Documentation Set

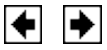
Links

ProActive Scheduler Toolbox

Getting Started

What Is ProActive Scheduler Toolbox?	An overview of the features, functions, and uses of ProActive Scheduler Toolbox.
Installation	Instructions on how to deploy the ProActive Scheduler for Matlab usage.
Starting And Connecting	Instructions on how to connect to ProActive Scheduler.
Monitoring	Instructions on how to use ProActive Scheduler monitoring tools.
Running Matlab functions remotely	Instruction on the use ProActive Scheduler Toolbox to run matlab code remotely.
Disconnected mode	Description of ProActive Scheduler Toolbox's disconnected mode.
Configuring PAsolve behavior and Debugging	How to configure ProActive Scheduler Toolbox behavior.
Using ProActive Scheduler Toolbox Simulink GUI	A detailed description of the use of ProActive Scheduler's Simulink GUI.

What Is ProActive Scheduler Toolbox?



What Is ProActive Scheduler Toolbox?

On this page…

Introduction

Can I Use ProActive Scheduler Toolbox?

Which versions of Matlab and Operating Systems are compatible?

What the current limitations of ProActive Scheduler Toolbox?

Introduction

ProActive Scheduler Toolbox is a matlab toolbox designed to provide interaction with the ***ProActive Scheduler***, part of ***ProActive Parallel Suite***.

ProActive Parallel Suite is an innovative Open Source solution (OW2) for parallel, distributed, multi-core computing.

ProActive features Java Parallel Programming seamlessly integrated with Scheduling and Resource Management. ProActive simplifies the programming and execution of parallel applications on Linux, Windows and Mac, together with the management of resources such as Desktop, Servers, Clusters, Enterprise GRIDs and Clouds.

The objectives of ***ProActive Scheduler Toolbox*** are to provide you with tools that:

- Allow you to run Matlab functions on remote computers.
- Do not block the local Matlab session while remote results are being produced.
- Allow you to seamlessly retrieve results when you need them, just as if the functions were run locally.
- Provide you detailed remote log/output information, altogether with errors if any occurred.
- Allow a disconnected mode, jobs can be submitted and the matlab session doesn't need to remain active while the job is processing.
- Allow automatic source transfer, data file transfer, transfer of local matlab workspace and other configurable options.

Back to Top

Can I Use ProActive Scheduler Toolbox?

ProActive Scheduler Toolbox is designed for both beginners and advanced users. There are basically two usage roles:

- The **administrator**: responsible for the toolbox installation on every machines, and the administration of ProActive Scheduler.
- The **toolbox user**: who will simply use the installed toolbox to run matlab code in parallel.

The basic requirements for the **administrator** is to have a standard Information Technology knowledge on windows or linux, for example:

- Start programs on windows or linux using the command line shell (sh or bat).
- Change environment variables on windows or linux.
- Be confident with XML syntax.
- Understand network protocols and principles.
- Have a basic knowledge of Security principles such as asymetric key pairs, and accounts management frameworks such as Ldap.
- Have a basic knowledge of the Java Programming Language.
- Have a basic knowledge of scripting languages such as Javascript, Python or Ruby.

The basic requirement for the **toolbox user** is to be able to write standard Matlab code. A few aspects must be kept in mind while designing functions that will be run remotely :

- Any reference to a file must be done using relative paths, excluding any use of '..' from the current folder (any tree hierarchy higher than the current directory has no meaning remotely).
- Files to be transferred to or from a remote worker must be declared explicitly through the InputFiles OutputFiles tasks information.
- Paths must be written with a portable syntax (usage of filesep() is recommended).
- Functions of any number of input parameters are accepted but number of output parameters must be 1.
- Functions must be independent from each others executions. It is not possible to write tightly coupled code with inter-dependant communications.

Back to Top

Which versions of Matlab and Operating Systems are compatible?

ProActive Scheduler Toolbox has been tested and is compatible with the following Matlab Versions and Operating Systems :

- Matlab 2007b, 2008b, 2009b and 2010b.
- Windows 32 and 64 bits (XP, Server, Seven).
- Linux 64bits.

On Windows 64bits, a patch described at SCHEDULING-1025 must be installed.

Back to Top

What the current limitations of ProActive Scheduler Toolbox?

Know bugs and limitations are reported on ProActive Scheduler's JIRA bug reporting tool:

https://bugs.activeeon.com/browse/SCHEDULING/component/10151

Back to Top

Getting Started Installation

Installation

On this page…

Introduction

Downloading and Installing ProActive Scheduler

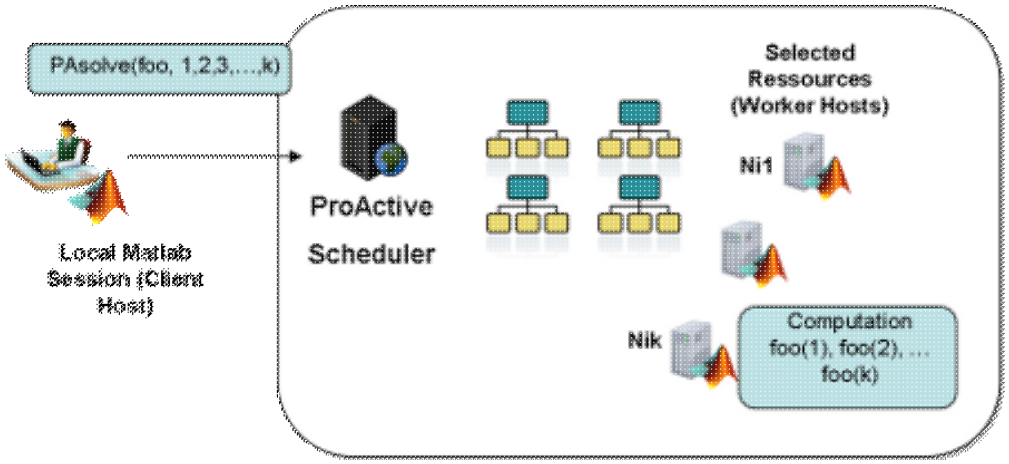
Scheduler Configuration For Matlab

Configuration of Remote Workers for Matlab

Setting up java.policy file

Introduction

A typical deployment of ProActive Scheduler with Matlab can be seen on the diagram below :



- The local Matlab session will connect to the ProActive Scheduler.
- The local user will submit a function **foo** to call with a set of parameters.
- The Scheduler will select among all its resources, those suited to the job.
- A Matlab engine will be started on each selected resource
- Each **foo(k)** will be executed on remote engines
- Results will be forwarded back to the user

Back to Top

Downloading and Installing ProActive Scheduler

ProActive Scheduler Toolbox can be downloaded from :

http://www.activeeon.com/community-downloads

The package which must be downloaded is called **ProActive Scheduling**
After downloading the package unzip it in a directory e.g. D:\Scheduling

IMPORTANT: According to your deployment infrastructure, ProActive Scheduler must be installed on :

1. The host which will launch the **Scheduler**.
2. Each **worker host** used by the Scheduler.
3. Each **client host** which will connect to the Scheduler to submit matlab functions.

The prerequisite for using ProActive is a **Java Virtual Machine**. On the above list, machines of group 1 and 2 must have a Java Virtual Machine installed and available on the Path of either cmd.exe for Windows or sh for linux. For many ProActive features it is also necessary to define the environment variable JAVA_HOME to the installation directory of the Java Virtual Machine.

If a worker host is running on Windows 64bits, a patch must be installed on top of ProActive installation, refer to SCHEDULING-1025 for details.

Complete configuration and installation of ProActive Scheduler is beyond the scope of this help, for more information, refer to:

ProActive Scheduler Manual

On each **client host**, on the matlab prompt add the path to ProActive Scheduler Toolbox (the folder SCHEDULING/extensions/matlab/toolbox):

```
addpath('D:\Scheduling\extensions\matlab\toolbox');
```

Back to Top

Scheduler Configuration For Matlab

On the host which will start the **Scheduler**, inside the file SCHEDULING/config/rm/settings.ini, the following settings must be entered:

```
pa.rm.select.script.timeout=60000

pa.rm.selection.maxthreadnumber=1
```

Back to Top

Configuration of Remote Workers for Matlab

On each **worker host** that will be used to compute remotely matlab functions, a specific configuration file must be written to tell the scheduler where to find a matlab installation. The configuration file is called **MatlabConfiguration.xml**, and is located in the folder **SCHEDULING/extensions/matlab/config/worker** for each **Worker Hosts**

By default, the **SCHEDULING/extensions/matlab/config/worker** folder only contains two template files one for Linux and one for Windows called **MatlabConfigurationTemplateUnix.xml** and **MatlabConfigurationTemplateWindows.xml**, one of this template must be edited and renamed into MatlabConfiguration.xml to setup the configuration for the current **Worker Host**

For example, here is MatlabConfigurationTemplateWindows.xml

```
<MatlabInstallations>
    <installation>
        <version>7.9</version>
        <home>C:\Program Files\MATLAB\R2009b</home>
        <libdir>bin\win32</libdir>
        <extdir></extdir>
        <bindir>bin</bindir>
        <command>matlab.exe</command>
        <ptolemydir>dist/lib/7.9/bin/win32</ptolemydir>
    </installation>
</MatlabInstallations>
```

Here is an explanation of tags used by this file:

- **version:** The matlab version installed in numeric format.
- **home:** Full path to matlab home directory.
- **libdir:** relative path (from homedir) to matlab dlls (bin\win32 for Windows 32, bin\win64 for Windows64.
- **extdir:** not used on Windows see contents of MatlabConfigurationTemplateLinux.xml for Linux hosts.
- **bindir:** relative path to the directory where to find the matlab executable.
- **command:** matlab executable name.
- **ptolemydir:** relative path (from ProActive Scheduler installation directory), where to find the native libraries for this Operating System and this Matlab version. The relative path is always dist/lib/$version/bin/$os, where:
    - $version is matlab version numeric version
    - $os is one of the following:
        - win32 for Windows 32bits
        - win64 for Windows 64bits
        - glnxa64 for Linux 64bits

The file supports multiple matlab configurations on the same Host (i.e. multiple matlab versions), and ProActive Scheduler Toolbox allows to select specific matlab versions for a given job dynamically. There is although an important limitation on Windows: Matlab engine on windows works as a registered COM service. The registration as a COM service works for only one installation instance(version) of Matlab. If other matlab engines that the one registered try to start, it will fail. There exists a workaround to this issue, it is possible to run the command "matlab /regserver" to register a specific version of Matlab as the current matlab engine, but Administrators or Super Users only can run the command. In case of multiple Matlab configurations specified on Windows, ProActive Scheduler Toolbox will try to run the "matlab /regserver", but the scheduler will have to be a Super User account for this to work.

To resume the problem on Windows, you can either:

- Have one single instance of Matlab installed on a given host.
- Have multiple instances of Matlab installed. Run "matlab /regserver" from an Administrator account on a specific matlab version and configure that version in MatlabConfiguration.xml.
- Have multiple instances of Matlab installed and let ProActive Scheduler run on a Administrator or Power User account (matlab /regserver commands will be run dynamically by ProActive Scheduler Toolbox).

Back to Top

Setting up java.policy file

On each **client host** that will connect to the Scheduler to submit tasks, it is mandatory to add a specific configuration file for Java called the **Java Policy** file. This file is meant to define the security settings of a given Java Virtual Machine. This configuration can either be :

- Specific to a given **matlab installation** (but common for all users on this machine).
- Specific to a **given user**, but common to all matlab installations of this machine (and also all other Java programs).

Matlab specific configuration

In order to do the first type of configuration, one needs to have write access to Matlab's installation directory (requires administrator privilege). First a file called **matlab.java.policy** must be created in the directory **MATLAB/bin/ARCH** where:

- **MATLAB** refers to your Matlab installation directory.
- **ARCH** refers to the architecture directory : glnax64 for linux 64bits, win32 for Windows 32bits and win64 for Windows 64bits.

The **matlab.java.policy** file must contain the following:

```
grant {
  permission java.security.AllPermission;
};
```

Finally, a second file called **java.opts** must be created in the same directory with the following content:

```
-Djava.security.manager \
-Djava.security.policy=C:\Program Files\MATLAB\R2007b\bin\win64\matlab.java.policy
```

Here **java.security.policy** points to the full path of the **matlab.java.policy** file, replace this path with yours.

These two files created, you can start matlab and verify that the configuration has been done correctly, by typing the following command on the matlab prompt:

```
>> java.lang.System.getProperty('java.security.policy')

ans =

C:\Program Files\MATLAB\R2007b\bin\win64\matlab.java.policy
```

You must see the path that you specified in the **java.opts** file. If you don't see this path, it means there is a syntax error in the java.opts file, try different things, like writing everything on one line, using a backslash to separate lines, etc (as the syntax accepted varies on different Operating Systems).

If the configuration has not been done properly, a call to PAconnect (see Chapter Connecting to a running ProActive Scheduler from Matlab) will result in displaying a big error message containing the following Exception:

```
java.security.AccessControlException: access denied (java.util.PropertyPermission org.mortbay.io.nio.JVMBU
```

Again recheck your configuration if this message appear.

User specific configuration

The second type of java policy configuration is simpler to do than the first one, but it will apply to every Java program started by this user on your machine, including Java applets. Accordingly, this set up should be done with care.

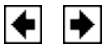
In order to do the second configuration, go to the **HOME** directory of the user that will use the toolbox. The **HOME** directory can be found by typing the following command on the matlab prompt:

```
>> java.lang.System.getProperty('user.home')

ans =

/user/fviale/home
```

In this HOME directory, create a file called **.java.policy** (notice the first dot in the file name), which will contain the same text as the **matlab.java.policy** file from the first method. That's it! Again, if the configuration is not done properly the same exception will appear when trying to connect to the Scheduler.

Back to Top

What is ProActive Scheduler Toolbox? Starting And Connecting



Starting and Connecting

On this page…

Introduction

Starting ProActive Scheduler

Connecting to a running ProActive Scheduler from Matlab

Introduction

ProActive Scheduler is in fact composed of two programs working together:

- **ProActive Resource Manager**: the resource manager is the program in charge of a pool of resources called Nodes. Each node is a Java Virtual Machine running on a worker host. There can be several of these Nodes deployed on a single host, generally when a host has multiple processes or cores. It allows to maximize the processing power available on the network. The Resource Manager needs to be configured according to the network topology in order to deploy Nodes on available machines. Configuration of the Resource Manager for acquiring resources is beyond the scope of this document and is explained on the Resource Manager manual available at http://proactive.inria.fr/release-doc/Resourcing/multiple_html/index.html.
- **ProActive Scheduler**: the scheduler receives jobs to be executed. It will schedule their execution according to its policy and its workload. When a job is ready to be scheduled, the scheduler will contact the resource manager to find the maximum number of resources available to execute the job. Resources will be selected sometimes according to specific policies. In case of ProActive Scheduler Toolbox, only resources which have a valid Matlab configuration will be selected. A lot of extra selection will be done dynamically, for example specific Matlab versions needed by the job, or toolbox tokens availability .

Back to Top

Starting ProActive Scheduler

Before starting ProActive Scheduler and Resource Manager, make sure that you:

- Changed the default Resource Manager settings according to the paragraph Scheduler Configuration For Matlab.
- Configured each Matlab worker host according to the paragraph Configuration of Remote Workers for Matlab.

The commands to start ProActive Scheduler are located in SCHEDULING/bin/unix or SCHEDULING/bin/windows, depending on your Operating System. The following commands are present:

- **rm-start.sh(.bat)**: To start ProActive Resource Manager.
- **rm-start-clean.sh(.bat)**: To start ProActive Resource Manager with a clean database (first startup).
- **rm-start.sh(.bat)**: To start ProActive Scheduler.
- **scheduler-start-clean.sh(.bat)**: To start ProActive Scheduler with a clean database (first startup).

The following sequence of commands (executed from SCHEDULING/bin/unix) will start a Resource Manager will 4 local Nodes(JVM) and a Scheduler. Although this trivial deployment is not meant to be a practical case, it is still a good way to test the framework and become familiar with ProActive.

First the Resource Manager:

```
 $ rm-start-clean -ln
Starting Resource Manager, Please wait...
Resource Manager successfully created on rmi://pendule.inria.fr:1099/
```

Then the Scheduler:

```
 $ scheduler-start-clean
Starting Scheduler, Please wait...
Resource Manager URL was not specified, connection made to the local Resource Manager at rmi://pendule.in
Starting scheduler...
Scheduler successfully created on rmi://pendule.inria.fr:1099/
```

Back to Top

Connecting to a running ProActive Scheduler from Matlab

From a matlab session, assuming that ProActive Scheduler Toolbox is already in Matlab path, run the following command (where PAconnect's argument is the url you received from the Scheduler starting command):

```
>> PAconnect('rmi://pendule.inria.fr:1099/');
```

A popup window will appear asking for a username and password. This username/password refers to the username and password of your account on ProActive Scheduler. ProActive Scheduler features a full account management facility along with the possibility to synchronize to existing Windows or Linux accounts via LDAP. More information can be found at Scheduler Manual:Configure users authentication.

Here for this example we will use the default account with login **demo** and password **demo**.

Here is what is displayed when the connection worked successfully:

```
>> PAconnect('rmi://pendule:1099/')
Connection successful, please enter login/password
Login succesful
Creating dataspace handler, please wait...
Dataspace handler created
>>
```

If the scheduler is unreachable, here is what is displayed :

```
>> PAconnect('rmi://pendule:1099/')
??? Java exception occurred:
org.ow2.proactive.scheduler.common.exception.ConnectionException: java.io.IOException: The url
rmi://pendule:1099/SCHEDULER is not bound to any known object
        at org.ow2.proactive.scheduler.common.SchedulerConnection.join(SchedulerConnection.java:102)
        at
        org.ow2.proactive.scheduler.ext.matsci.client.AOMatSciEnvironment.join(AOMatSciEnvironment.java:2
                at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

```
        at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
        at java.lang.reflect.Method.invoke(Unknown Source)
        at org.objectweb.proactive.core.mop.MethodCall.execute(MethodCall.java:395)
        at org.objectweb.proactive.core.body.request.RequestImpl.serveInternal(RequestImpl.java:253)
        at org.objectweb.proactive.core.body.request.RequestImpl.serve(RequestImpl.java:197)
        at
        org.objectweb.proactive.core.body.BodyImpl$ActiveLocalBodyStrategy.serveInternal(BodyImpl.java:61
                at
        org.objectweb.proactive.core.body.BodyImpl$ActiveLocalBodyStrategy.serve(BodyImpl.java:577)
        at org.objectweb.proactive.core.body.AbstractBody.serve(AbstractBody.java:944)
        at org.objectweb.proactive.Service.serveOldest(Service.java:214)
        at
        org.ow2.proactive.scheduler.ext.matsci.client.AOMatSciEnvironment.runActivity(AOMatSciEnvironment
                at org.objectweb.proactive.core.body.ActiveBody.run(ActiveBody.java:198)
        at java.lang.Thread.run(Unknown Source)
Caused by: java.io.IOException: The url rmi://pendule:1099/SCHEDULER is not bound to any known object
        at org.objectweb.proactive.api.PAActiveObject.lookupActive(PAActiveObject.java:1524)
        at org.ow2.proactive.authentication.Connection.lookupAuthentication(Connection.java:94)
        at org.ow2.proactive.authentication.Connection.connect(Connection.java:105)
        at org.ow2.proactive.scheduler.common.SchedulerConnection.join(SchedulerConnection.java:100)
        ... 15 more

Error in ==> PAconnect at 83
    ok = solver.join(url);
```

Back to Top

Installation

Monitoring

Monitoring

On this page…

Introduction

Scheduler and Resource Manager command line controllers

Scheduler and Resource Manager GUI

Introduction

The present chapter is a quick start guide to **ProActive Scheduler** and **Resource Manager** controllers and Graphical User Interfaces. These tools allow to execute a collection of operations on the Scheduler or Resource Manager, but in the context of **ProActive Scheduler Toolbox**, we will be interested mostly in the monitoring part.

Back to Top

Scheduler and Resource Manager command line controllers

The command line controllers of **ProActive Scheduler** and **Resource Manager** are called respectively **scheduler-client** and **rm-client**. They are located in SCHEDULING/bin/unix or SCHEDULING/bin/windows depending on your operating system.

- **scheduler-client** allows to submit jobs to **ProActive Scheduler** in XML format, monitor job executions, preview results, view job logs, etc.
- **rm-client** allows to deploy or monitor resources in **Resource Manager**.

Here is an example use of scheduler-client, default login and password is **demo/demo**:

```
fviale@pendule unix $ scheduler-client
Trying to connect Scheduler on rmi://localhost/
        -> Connection established on rmi://localhost/

Connecting client to the Scheduler
login: demo
password: ****
Retrieved public key from Scheduler at rmi://localhost/
        -> Client 'demo' successfully connected

Type command here (type '?' or help() to see the list of commands)


 > listjobs();
     ID      NAME                          OWNER      PRIORITY     PROJECT          STATUS       START AT

     1       Matlab Environment Job 0      demo       Normal       Not Assigned     Finished     18:33:53  (
     2       Matlab Environment Job 1      demo       Normal       Not Assigned     Finished     20:50:33  (
     3       Matlab Environment Job 2      demo       Normal       Not Assigned     Finished     20:51:39  (
```

```
     4      Matlab Environment Job 3      demo      Normal      Not Assigned     Finished     20:52:25  (
     5      Matlab Environment Job 4      demo      Normal      Not Assigned     Finished     20:53:10  (
     6      Matlab Environment Job 5      demo      Normal      Not Assigned     Finished     20:54:38  (
     7      Matlab Environment Job 6      demo      Normal      Not Assigned     Finished     20:57:46  (
     8      Matlab Environment Job 7      demo      Normal      Not Assigned     Finished     23:07:09  (
     9      Matlab Environment Job 8      demo      Normal      Not Assigned     Finished     01:05:02  (
     10     Matlab Environment Job 9      demo      Normal      Not Assigned     Finished     01:08:09  (
     11     Matlab Environment Job 10     demo      Normal      Not Assigned     Finished     01:11:35  (
     12     Matlab Environment Job 11     demo      Normal      Not Assigned     Finished     01:20:11  (
     13     Matlab Environment Job 12     demo      Normal      Not Assigned     Finished     01:56:33  (
     14     Matlab Environment Job 13     demo      Normal      Not Assigned     Finished     02:20:19  (
     15     Matlab Environment Job 14     demo      Normal      Not Assigned     Finished     02:21:03  (
     16     Matlab Environment Job 0      demo      Normal      Not Assigned     Finished     02:40:00  (

 > jobstate(15)

   Job '15'    name:Matlab Environment Job 14    owner:demo    status:Finished    #tasks:20

     ID       NAME   ITER   DUP   STATUS     HOSTNAME                          EXEC DURATION   TOT DURATIO

     150001   9_0                 Finished   pendule.inria.fr (PA_JVM41...    733ms            2s 943ms
     150002   5_0                 Finished   pendule.inria.fr (PA_JVM15...    753ms            3s 601ms
     150003   8_0                 Finished   pendule.inria.fr (PA_JVM19...    637ms            2s 531ms
     150004   10_0                Finished   pendule.inria.fr (PA_JVM15...    723ms            1s 734ms
     150005   1_0                 Finished   pendule.inria.fr (PA_JVM41...    635ms            2s 493ms
     150006   4_0                 Finished   pendule.inria.fr (PA_JVM14...    823ms            3s 708ms
     150007   16_0                Finished   pendule.inria.fr (PA_JVM15...    803ms            2s 405ms
     150008   7_0                 Finished   pendule.inria.fr (PA_JVM19...    733ms            2s 953ms
     150009   6_0                 Finished   pendule.inria.fr (PA_JVM14...    626ms            3s 741ms
     150010   2_0                 Finished   pendule.inria.fr (PA_JVM41...    724ms           798ms
     150011   0_0                 Finished   pendule.inria.fr (PA_JVM14...    612ms            3s 553ms
     150012   3_0                 Finished   pendule.inria.fr (PA_JVM14...    832ms            3s 575ms
     150013   17_0                Finished   pendule.inria.fr (PA_JVM15...    614ms            3s 631ms
     150014   15_0                Finished   pendule.inria.fr (PA_JVM19...    637ms            2s 490ms
     150015   19_0                Finished   pendule.inria.fr (PA_JVM41...    716ms            2s 532ms
     150016   11_0                Finished   pendule.inria.fr (PA_JVM15...    621ms            3s 675ms
- more - (q : abort | a : display all | any : next page)
     150017   12_0                Finished   pendule.inria.fr (PA_JVM19...    615ms           687ms
     150018   13_0                Finished   pendule.inria.fr (PA_JVM14...    617ms           596ms
     150019   18_0                Finished   pendule.inria.fr (PA_JVM19...    625ms            2s 487ms
     150020   14_0                Finished   pendule.inria.fr (PA_JVM41...    720ms            2s 502ms

 > task

taskoutput(   taskresult(
 > taskresult(15,'12_0')
Task 12_0 result =>
     true

 > taskoutput(15,'12_0')
12_0 output :
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
```

```
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:24 CET 2011 ][pendule OUT]

    > exit();
   Exiting controller.
```

The **help()** function gives a description of all available comands. More information about the scheduler controller can be found at Using the Scheduler controller.

Here is an example use of rm-client:

```
fviale@pendule unix $ rm-client
Trying to connect RM on rmi://localhost:1099/
        -> Connection established on rmi://localhost:1099/

Connecting to the RM
login: demo
password: ****
Retrieved public key from Resource Manager at rmi://localhost:1099/
        -> Client 'demo'  successfully connected

Type command here (type '?' or help() to see the list of commands)


 > listnodes();
     SOURCE NAME     HOSTNAME           STATE   SINCE            URL

     GCMLocalNodes   pendule.inria.fr   Free    13/01/11 02:40   rmi://pendule.inria.fr:6608/PA_JVM148585
     GCMLocalNodes   pendule.inria.fr   Free    13/01/11 02:40   rmi://pendule.inria.fr:6608/PA_JVM157471
     GCMLocalNodes   pendule.inria.fr   Free    13/01/11 02:40   rmi://pendule.inria.fr:6608/PA_JVM191614
     GCMLocalNodes   pendule.inria.fr   Free    13/01/11 02:40   rmi://pendule.inria.fr:6608/PA_JVM416643

 >
```

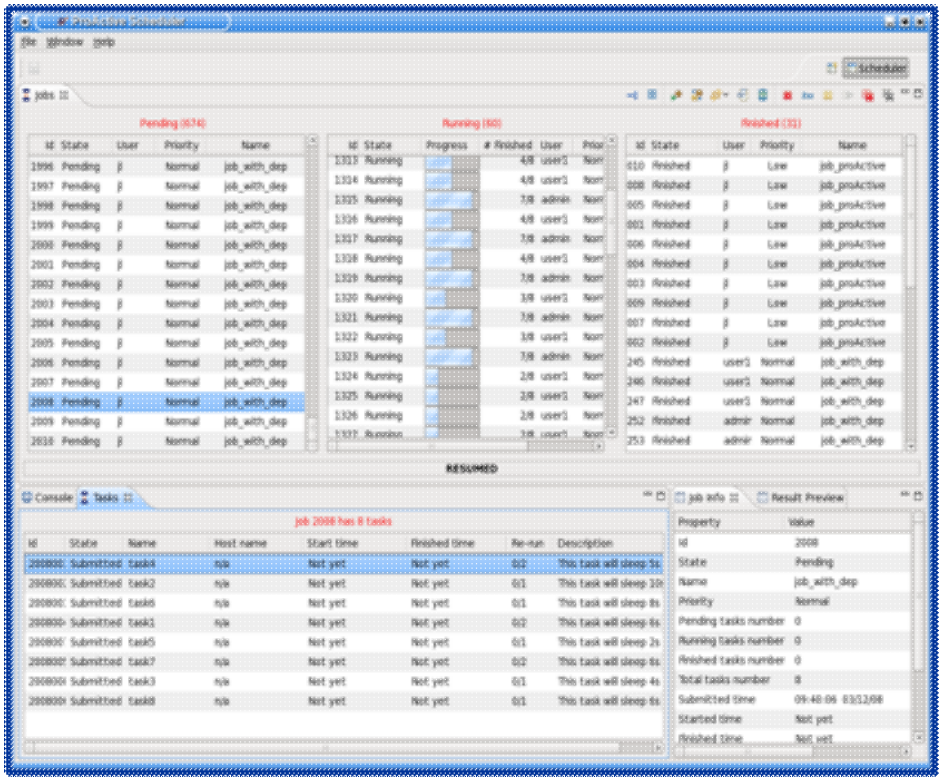
Here we see that the Resource Manager has been deployed with 4 local nodes. The STATE column displays if the node is currently executing a job, free or there was a node failure. More information about the Resource Manager controller can be found at Interacting with the resource manager.

Back to Top

Scheduler and Resource Manager GUI

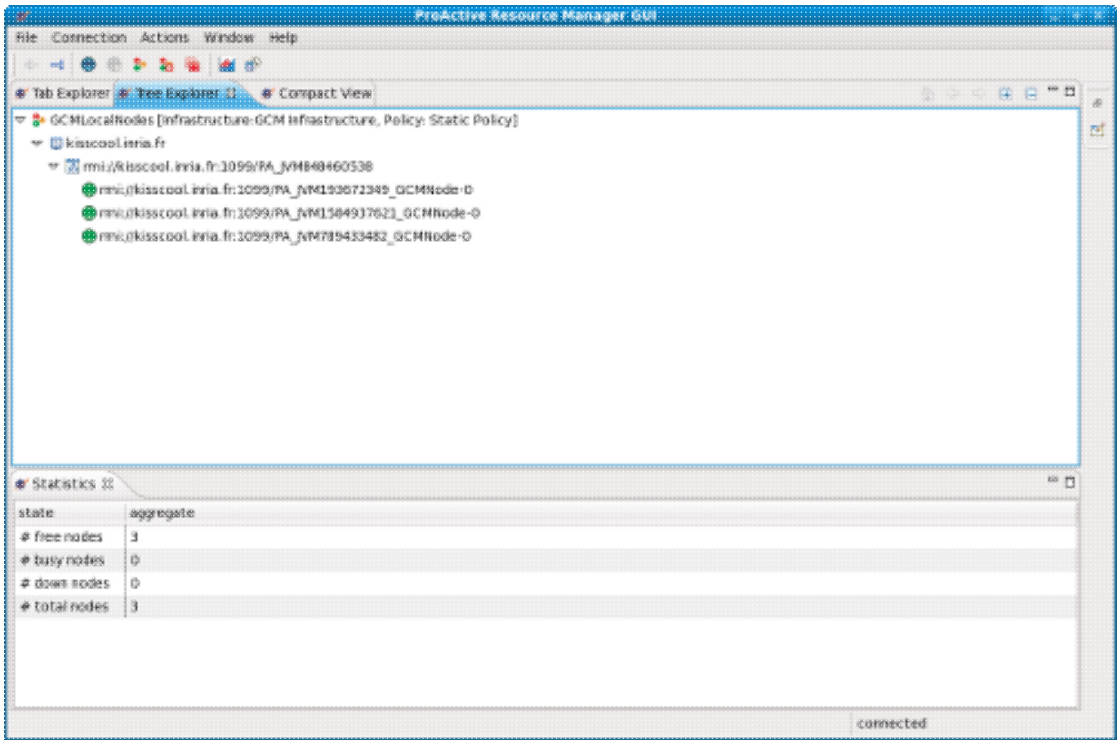
Scheduler and Resource Manager come as well with graphical user interfaces which provide even more features than the command line interfaces, in a graphical and intuitive way. The GUI must be downloaded separately from ProActive Scheduler on the ProActive downloads page. They required a Java Virtual Machine installed in order to run.

Here is a view of the **Scheduler GUI**:



Information on its usage can be found at ProActive Scheduler Eclipse Plugin.

Here is a view of the **Resource Manager GUI**:

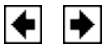


Information on its usage can be found at Resource Manager Graphical User Interface.

Back to Top

Starting And Connecting

Running Matlab functions remotely



Running Matlab functions remotely

On this page…

Introduction

Simplest parametric sweep

Parametric sweep with user code

Receiving results

Adding Input and Output Files

Chaining Remote Tasks

Introduction

The three main actors of ProActive Scheduler Toolbox are the following functions and objects:

- **PAsolve**: It is the function used to run matlab code remotely. A call to PAsolve will create a Matlab job inside ProActive Scheduler. After the job is created, the PAsolve function will return, allowing the local Matlab session to continue while the results are being produced.
- **PAResult**: this object is returned by the PAsolve function. PAsolve returns a vector of PAResult objects, whose size matches the number of parameters to the PAsolve function. The PAResult object defines a collection of methods which can be used to wait (block the local Matlab session) for specific results.
- **PATask**: this object allows to define complex tasks that can be given as parameters to PAsolve.

Back to Top

Simple parametric sweep

In this paragraph, we will explain how PAsolve should be called to evaluate remotely a matlab function. It is possible to do a single remote evaluation, but here we will use as example a multiple remote evaluation of the same function but with different parameters (parametric sweep). For this example, we will use the matlab function **factorial**, which has no practical use to be run remotely, but which will serve as a well-known example. After the matlab session is connected to the scheduler (explained in paragraph Connecting to a running ProActive Scheduler from Matlab), enter the following command on the matlab prompt:

```
>> res=PAsolve(@factorial,1,2,3,4,5)
Job submitted : 1
Awaited (J:1)
Awaited (J:1)
Awaited (J:1)
Awaited (J:1)
Awaited (J:1)
```

PAsolve is being called with a function handle to the matlab function "factorial". A list of parameters from 1 to 5 is being given. This means that remotely factorial(1), factorial(2),..., factorial(5) will be executed. The PAsolve call returns immediately, giving the ID of the job created (1) inside ProActive Scheduler and returning in the res variable an array of PAresult objects which display themselves as *Awaited* objects from job ID 1.

After a while, factorial results will be produced. It is possible to force Matlab to wait(block) for these results using a specific function call (explained in next chapter: Receiving results). But here, we will simply evaluate the variable **res** until it displays itself as received :

```
>> res
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [StartPARuntime] Starting a ProActiveRuntime on pendu
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Logging to org.slf4j.impl.Log4jLoggerAdapt
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] jetty-6.1.x
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Started SelectChannelConnector@0.0.0.0:530
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RegistryHelper] Detected an existing RMI Registry on
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [ProActiveRuntimeImpl] unable to activate RTBroadcast
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]


res =

     1

[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [StartPARuntime] Starting a ProActiveRuntime on pendu
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Logging to org.slf4j.impl.Log4jLoggerAdapt
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] jetty-6.1.x
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Started SelectChannelConnector@0.0.0.0:520
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [RegistryHelper] Detected an existing RMI Registry on
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [ProActiveRuntimeImpl] unable to activate RTBroadcast
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
```

```
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]


res =

     2

[ Tue Jan 11 14:27:34 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:34 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]


res =

     6

[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [StartPARuntime] Starting a ProActiveRuntime on pendu
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Logging to org.slf4j.impl.Log4jLoggerAdapt
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] jetty-6.1.x
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Started SelectChannelConnector@0.0.0.0:51
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object 
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object 
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object 
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object 
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RegistryHelper] Detected an existing RMI Registry o
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [ProActiveRuntimeImpl] unable to activate RTBroadcas
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]


res =

    24

[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [StartPARuntime] Starting a ProActiveRuntime on pendu
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Logging to org.slf4j.impl.Log4jLoggerAdapt
```

```
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] jetty-6.1.x
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Started SelectChannelConnector@0.0.0.0:52
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RegistryHelper] Detected an existing RMI Registry o
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [ProActiveRuntimeImpl] unable to activate RTBroadcas
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]


res =

   12
```

This output shows the remote outputs of the five executions factorial(1..5). The beginning is an output message related to ProActive internal behavior which basically says that ProActive was successfully started on this remote Resource (and the associated Matlab Engine). This output message only appears at the beginning of a ProActive with Matlab deployment, when a remote Node is first used for a Matlab Task. The rest doesn't show any matlab output, as the factorial function doesn't show any.

The remote outputs are displayed only once. They can be accessed later using the **logs** attribute of the PAResult task:

```
>> res

res =

     1


res =

     2


res =

     6


res =

    24


res =
```

```
   120

>> res.logs

ans =

[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [StartPARuntime] Starting a ProActiveRuntime on pendu
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Logging to org.slf4j.impl.Log4jLoggerAdapt
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] jetty-6.1.x
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Started SelectChannelConnector@0.0.0.0:53
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object 
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object 
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object 
[ Tue Jan 11 14:27:29 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object 
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RegistryHelper] Detected an existing RMI Registry o
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [ProActiveRuntimeImpl] unable to activate RTBroadcas
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]



ans =

[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [StartPARuntime] Starting a ProActiveRuntime on pendu
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Logging to org.slf4j.impl.Log4jLoggerAdapt
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] jetty-6.1.x
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Started SelectChannelConnector@0.0.0.0:52
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object 
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object 
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object 
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object 
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [RegistryHelper] Detected an existing RMI Registry o
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [ProActiveRuntimeImpl] unable to activate RTBroadcas
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
```

```
ans =

[ Tue Jan 11 14:27:34 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:34 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:35 CET 2011 ][pendule OUT]



ans =

[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [StartPARuntime] Starting a ProActiveRuntime on pendu
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Logging to org.slf4j.impl.Log4jLoggerAdapt
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] jetty-6.1.x
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Started SelectChannelConnector@0.0.0.0:511
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RegistryHelper] Detected an existing RMI Registry or
[ Tue Jan 11 14:27:31 CET 2011 ][pendule OUT][INFO ] [ProActiveRuntimeImpl] unable to activate RTBroadcast
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]



ans =

[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [StartPARuntime] Starting a ProActiveRuntime on pendu
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Logging to org.slf4j.impl.Log4jLoggerAdapt
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] jetty-6.1.x
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [Slf4jLog] Started SelectChannelConnector@0.0.0.0:52
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RemoteObjectProtocolFactoryRegistry] Remote Object F
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [RegistryHelper] Detected an existing RMI Registry or
[ Tue Jan 11 14:27:30 CET 2011 ][pendule OUT][INFO ] [ProActiveRuntimeImpl] unable to activate RTBroadcast
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
```

```
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:32 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
[ Tue Jan 11 14:27:33 CET 2011 ][pendule OUT]
```

But running the factorial function is not interesting practically, in the next chapter Parametric sweep with user code, we will described a more useful scenario.

Back to Top

Parametric sweep with user code

In this chapter, we will create user-defined matlab scripts that we will try to run remotely. Here are the two functions that we will write:

- A function called **helloworld** which will not compute anything but will try to display remotely the message "Hello World!".
- A function called **makeerror** which will produce a matlab error remotely.

Here is how the **helloworld.m** file is defined :

```
function ok=helloworld()
disp('HelloWorld');
ok=true;
```

Here is the PAsolve execution of **helloworld**:

```
>> res=PAsolve(@helloworld,{},{},{})
Job submitted : 2
Awaited (J:2)
Awaited (J:2)
Awaited (J:2)
>> res
[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]>> HelloWorld
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
```

```
res =

     1

[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]>> HelloWorld
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]


res =

     1

[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]>> HelloWorld
[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:02 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]
[ Tue Jan 11 16:30:03 CET 2011 ][pendule OUT]


res =

     1
```

You see in this example how the HelloWorld string is displayed on the remote engines and appears in the job logs. Note as well the syntax of the call PAsolve(@helloworld,{},{},{}), which says that we call helloworld with no parameter (an empty cell array). If helloworld had more than one parameter, we would as well use a cell array to hold the parameters such as PAsolve(@multparam,{param_1_1,param_1_2,..param_1_k},...,{param_n_1,param_n_2,..param_n_k}). This would call the k-parameter function "multparam" n-times. In the rare case when the function requires a single parameter which is a cell, the following syntax should be used : PAsolve(@onecellfunc,{cellparam_1}, ..., {cellparam_n} , where cellparam_1..n are cells.

Here is how the **makeerror.m** file is defined :

```
function ok=makeerror()
% b doesn't exist!
disp(b)
```

```
ok=true;
```

Here is the PAsolve execution of **makeerror**:

```
>> res=PAsolve(@makeerror,{},{},{})
Job submitted : 4
Awaited (J:4)
Awaited (J:4)
Awaited (J:4)
>> res
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]??? Undefined function or variable 'b'.
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]Error in ==> makeerror at 3
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]disp(b)
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR] ??? Error using ==> save
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]Variable 'out' not found.
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:46 CET 2011 ][pendule OUT]


Error using ==> PAResult.PAwaitFor at 99
Error during remote script execution

Error in ==> PAResult.display at 49
            dp(PAwaitFor(RR), inputname(1))

[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]??? Undefined function or variable 'b'.
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]Error in ==> makeerror at 3
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]disp(b)
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR] ??? Error using ==> save
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]Variable 'out' not found.
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
```

```
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]


Error using ==> PAResult.PAwaitFor at 99
Error during remote script execution

Error in ==> PAResult.display at 49
            dp(PAwaitFor(RR), inputname(1))

[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]??? Undefined function or variable 'b'.
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]Error in ==> makeerror at 3
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]disp(b)
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR] ??? Error using ==> save
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]Variable 'out' not found.
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule ERR]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:45 CET 2011 ][pendule OUT]
[ Tue Jan 11 18:30:46 CET 2011 ][pendule OUT]


Error using ==> PAResult.PAwaitFor at 99
Error during remote script execution

Error in ==> PAResult.display at 49
            dp(PAwaitFor(RR), inputname(1))
```

You see in this example how the matlab errors from the remote execution are forwarded and appear in the logs.

Back to Top

Receiving results

As explained above, PAsolve calls are asynchronous and don't block Matlab session, a collection of functions are provided to wait(block) for results arrival or test their presence without blocking:

- **PAwaitAll**: given a vector of PAResult, blocks the matlab session until all results are received.
- **PAwaitAny**: given a vector of PAResult, blocks the matlab session until any one of those results are received. Successive calls to PAwaitAny, allows to retrieve the results one by one, in completion order.
- **PAisAwaited**: given a vector of PAResult, this non-blocking call tells which results are available.
- **PAResult.val attribute**: similarly to PAwaitAll, block the matlab session until all results are received. As it is an attribute, it provides full control on which particular results needs to be waited.

Here is an example of using **PAwaitAll**:

```
>> res=PAsolve(@factorial,1,2,3,4,5)
Job submitted : 2
Awaited (J:2)
Awaited (J:2)
Awaited (J:2)
Awaited (J:2)
Awaited (J:2)
>> val=PAwaitAll(res)
[ Tue Jan 11 20:19:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:09 CET 2011 ][pendule OUT]

[ Tue Jan 11 20:19:06 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:06 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:06 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:06 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]

[ Tue Jan 11 20:19:06 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:06 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]

[ Tue Jan 11 20:19:06 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:06 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
```

```
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]

[ Tue Jan 11 20:19:06 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:06 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:19:07 CET 2011 ][pendule OUT]


val =

    [1]    [2]    [6]    [24]    [120]

>> class(val)

ans =

cell

>> class(val{1})

ans =

double

>> class(res)

ans =

PAResult
```

Notice that **PAwaitAll** returns its result in a cell array, which contain the real result to the factorial call. The "res" variable above will always be of class **PAResult**, and thus is not usable directly.

Here is an example of using **PAwaitAny**:

```
>> res=PAsolve(@factorial,1,2,3,4,5)
Job submitted : 3
Awaited (J:3)
Awaited (J:3)
Awaited (J:3)
Awaited (J:3)
Awaited (J:3)
>> val1=PAwaitAny(res)
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
```

```
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]


val1 =

     1

>> val2=PAwaitAny(res)
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]


val2 =

     2

>> val3=PAwaitAny(res)
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]


val3 =

     6

>> val4=PAwaitAny(res)
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]
```

```
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:09 CET 2011 ][pendule OUT]


val4 =

    24

>> val5=PAwaitAny(res)
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:07 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]
[ Tue Jan 11 20:24:08 CET 2011 ][pendule OUT]


val5 =

   120

>> val6=PAwaitAny(res)
??? Error using ==> PAResult.PAwaitAny at 54
All results have already been accessed

>>
```

Notice how successive calls to **PAwaitAny** returns different answers, and how the last call produces an error as all results have already been received. Please note as well, that the order in which results are received by a **PAwaitAny** call does NOT necessary match the order of the **PAsolve** call (i.e. first calls returns 1, second 2, etc...). It only depends on the order of completion and not submission!

Finally, an example of using **PAisAwaited**:

```
>> res=PAsolve(@factorial,1,2,3,4,5)
Job submitted : 5
Awaited (J:5)
Awaited (J:5)
Awaited (J:5)
Awaited (J:5)
Awaited (J:5)
>> val=PAisAwaited(res)

val =

     1     1     1     1     1
```

```
>> val=PAisAwaited(res)

val =

     1     1     1     1     1


>> val=PAisAwaited(res)

val =

     1     1     1     0     1

>> val=PAisAwaited(res)

val =

     0     0     0     0     0

>>
```

We see in this example how results are progressively received. Finally the **val** attribute of the PAResult class can be used like PAwaitAll:

```
>> res=PAsolve(@factorial,1,2,3,4,5)
Job submitted : 1
Awaited (J:1)
Awaited (J:1)
Awaited (J:1)
Awaited (J:1)
Awaited (J:1)
>> res(1:2).val

[ Wed Jan 12 18:33:58 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:58 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:58 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:58 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:58 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:58 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:58 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:58 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:58 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]


[ Wed Jan 12 18:33:58 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:58 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
```

```
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]


ans =

    [1]    [2]

>> res(3).val
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:01 CET 2011 ][pendule OUT]


ans =

     6

>> res(4:5).val

[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:00 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:34:00 CET 2011 ][pendule OUT]


[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
[ Wed Jan 12 18:33:59 CET 2011 ][pendule OUT]
```

```
ans =

    [24]    [120]
```

You see how the **val** attribute has been used here to wait for the first two results, the third one and then the last two. You see as well how the variable type returned in the ans variable varies. Here is the convention : if more than one results is retrieved, the **val** attribute packs all results retrieved in a cell array. If only one result is retrieved, the **val** attribute returns it without packing.

We've seen above that we can as well call the **logs** attribute to retrieve task logs. Similarly a call to the **logs** attribute will block the matlab session until the result is available, and will behave the same as the **val** attribute concerning packing.

Back to Top

Adding Input and Output Files

Running single matlab functions remotely is the standard way of using **PAsolve**, but sometimes Matlab functions will also need to read input files and/or write output files. In this chapter, we will describe how to do that using the object class **PATask**.

The **PATask** object works exactly like a structure. It contains the following fields:

- **Func**:

  the function to be executed in this task, i.e. @factorial.
- **Params**: the parameters to the function, which can either be a single parameter or a cell array containing the function multiple parmaeters.
- **InputFiles**: the input files, given as a cell array of strings. Paths to the input files must be given relatively to the current directory and it is not allowed to refer to a file which is outside the current directory hierarchy (i.e. no '..').
- **OutputFiles**: the output files, given as a cell array of strings. Similarly the paths must be relative to the base directory of the remote matlab engine (which will be a subdirectory of the TEMP directory). The matlab function will have to create those files and all subdirectories relative to the base one.
- **Description**: a char array representing the task.
- **Compose**: true or false. When composing tasks, is the result of the previous task given as first parameter of this one (see next chapter Chaining Remote Tasks) ?

To illustrate the usage of **PATask**, we will define a simple Matlab function **factfile** which will read an integer from a mat file and will output the factorial of this integer to a second mat file

Here is the content of **factile.m**:

```
function ok=factfile()
load('-mat','factfile_in.m')
b=factorial(a);
save('factfile_out.m','b');
ok=true;
```

Below is the PAsolve execution of **factfile** with the creation of the PATask and input/output files. We execute only one task, as multiple task would involve handling multiple files and would be too complicate for this example. The PAsolve funcion takes the PATask as parameter, which contains all the necessary information:

```
>> t=PATask;
>> t.Func = @factfile;
```

```
>> t.Params = {};
>> t.InputFiles={'factfile_in.m'};
>> t.OutputFiles={'factfile_out.m'};
>> t

t(1,1) =

    Func:       @factfile
    Params:
    InputFiles: 'factfile_in.m'
    OuputFiles: 'factfile_out.m'
    Compose:       false
>> r=PAsolve(t)
Job submitted : 7
Awaited (J:7)
>> r.val
[ Wed Jan 12 20:57:46 CET 2011 ][pendule OUT]
[ Wed Jan 12 20:57:46 CET 2011 ][pendule OUT]
[ Wed Jan 12 20:57:47 CET 2011 ][pendule OUT]
[ Wed Jan 12 20:57:47 CET 2011 ][pendule OUT]
[ Wed Jan 12 20:57:47 CET 2011 ][pendule OUT]
[ Wed Jan 12 20:57:47 CET 2011 ][pendule OUT]
[ Wed Jan 12 20:57:47 CET 2011 ][pendule OUT]
[ Wed Jan 12 20:57:47 CET 2011 ][pendule OUT]
[ Wed Jan 12 20:57:47 CET 2011 ][pendule OUT]
[ Wed Jan 12 20:57:47 CET 2011 ][pendule OUT]
[ Wed Jan 12 20:57:47 CET 2011 ][pendule OUT]
[ Wed Jan 12 20:57:47 CET 2011 ][pendule OUT]
[ Wed Jan 12 20:57:47 CET 2011 ][pendule OUT]


ans =

     1

>> load('-mat','factfile_out.m')
>> disp(b)
   120
```

Back to Top

Chaining Remote Tasks

In the previous chapter, we learnt how to create a **PATask** with InputFiles and OutputFiles parameters. In this chapter, we will learn how to chain PATasks that will be run successively, i.e. the output of the first PATask will be given as the input of the second PATask, etc.

For the computation, we will use the matlab function sqrt which computes the square root of its argument. We won't use input or output files in this example, but of course it's possible to use this feature while chaining tasks:

```
>> t=PATask;
>> t.Func = @sqrt;
>> t.Params={2};
>> t(2,1)=t;
Warning: Number of parameters differs from function's number of inputs
> In PATask.PATask at 91
  In PATask.subsasgn at 72
```

```
>> t(2,1).Params={};
>> t(2,1).Compose=true;
>> t(3,1)=t(2,1);
>> t

t(1,1) =

    Func:       @sqrt
    Params: [2]
    Compose:       false

t(2,1) =

    Func:       @sqrt
    Params:
    Compose:       true

t(3,1) =

    Func:       @sqrt
    Params:
    Compose:       true
>> r=PAsolve(t)
Job submitted : 8
Awaited (J:8)
>> r.val
[ Wed Jan 12 23:07:13 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:13 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:14 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:14 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:14 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:14 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:14 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:14 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:14 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:14 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:14 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:14 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:14 CET 2011 ][pendule OUT]
[ Wed Jan 12 23:07:14 CET 2011 ][pendule OUT]


ans =

    1.0905

>> sqrt(sqrt(sqrt(2)))

ans =

    1.0905
```

Explanation: the first 3 lines create a **PATask** with function **sqrt** and parameter **2**. The fourth line replicates this task to the second line. While replicating, a warning is displayed because **PATask** is confused concerning the number of parameters of function **sqrt** and the number of parameters given (during the replication, the number of parameters is 0). These warnings are only meant to avoid mistakes, here we chose to ignore it. On the fifth and sixth command, we set the parameters to the second task to the empty list and we activate the **Compose** flag. This means that the fist argument of the function **Func** will be taken from the result of the previous task instead of the task **Params** attribute. As sqrt has only one parameter, we set the Params attribute to the empty list. Finally, on the seventh line we replicate the line two of the **PATask** matrix, without changing it. At

the end we verify that the result received matches the same computation done locally.

In these two examples we saw how to submit a PATask matrix, each time with a number of columns equal to 1. If we add more columns to the PATask matrix, it will mean that more parallel tasks will be submitted. To resume:

- A **line** vector of PATask of length k means that k PATask will be run in parallel
- A **column** vector of PATask of length m means that m PATask will be chained
- A **matrix** of PATask of size m,k means that k parallel series of m PATask chained together will be run

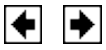
For convenience, it is also possible to call PAsolve with the following syntax:

```
r=PAsolve(c1,c2,...,ck)
```

where ci are column PATask vectors.

Back to Top

Monitoring Disconnected Mode



Configuring PAsolve behavior and Debugging

On this page…

Introduction

The PAoptions.ini file

Using the PAoptions function

Debugging

Introduction

PAsolve can be configured using various options. It can be done either:

- **Statically** by editing a file called **PAoptions.ini** which will be loaded when the toolbox is loaded.
- **Dynamically** by calling the **PAoptions** function which will affect every subsequent PAsolve calls.

Back to Top

The PAoptions.ini file

The **PAoptions.ini** file is located inside the ProActive Scheduler release at the following location :

```
SCHEDULING/extensions/matlab/config/toolbox/PAoptions.ini
```

It contains the following options:

- **Debug**: activate this option if any problem occurs and you want ProActive Scheduler Toolbox to run in debug mode (with verbose output)
- **TimeStamp**: activate this option in addition to Debug if you want that every debug line contains as well a Timestamp indicating the current system time.
- **TransferSource**: this option is by default set to on. It allows the automatic transfer of local source files used by matlab functions given to PAsolve to remote matlab engines. Turning this option to off is not recommended.
- **TransferEnv**: this option is by default set to off. It allows the workspace where the PAsolve call is done to be automatically transfered to the remote matlab engines. Global variables or variables existing in workspaces higher than the current one will not be transferred though.
- **TransferVariables**: this option is by default set to on. It allows the automatic transfer of parameters and return values via a mat file. Turning this option to off will let the transfer be done by the ptolemy java2matlab interface which faces numerical truncature issues. Turning this option to off is therefore not recommended.
- **CustomDataspaceURL** and **CustomDataspacePath**: by default these options are not set. Dataspaces are a ProActive internal mechanism allowing a shared file system between hosts. The mechanism allows the automatic transfer of files between the local matlab session and the remote engines. **ProActive Scheduler Toolbox** has an internal automatic

Dataspace creation. But sometimes, one would like to rely on an external Dataspace such as a ftp server. This Dataspace MUST be accessible from the local file system in order to work with **ProActive Scheduler Toolbox**.The **CustomDataspaceURL** option gives the Universally accessible URL of this dataspace and the **CustomDataspacePath** gives the path to access this dataspace from the local file system.

- **VersionPref**, **VersionMin**, **VersionMax** and **VersionRej**: these options allow the user to define which version of Matlab should/should not be used. In an heterogeneous environment, several versions of matlab can be available simultaneously on the network. These options ensures that the submitted matlab code will meet the correct matlab version. It is the responsibility of the user to find which versions are/are not compatible with the code.
- **VersionPref**: is the preferred version to use. It won't prevent another version to be used if the preferred version is not available.It is set automatically to the version of the local matlab session, but can be as well user defined via this option.
- **VersionMin** and **VersionMax**: is the minimum/maximum version to use. It is set by default to the minimum/maximum version supported.
- **VersionRej**: is a list of versions to be rejected. It is set by default to the versions for which the ptolemy java2matlab interface has not been compiled (and thus are not supported). These versions are all "a" releases matlab2007a, matlab2008a, etc...
- **TransferMatFileOptions**: this is the options used by the toolbox when generating mat files (in TransferEnv or TransferVariables). By default, it is set to '-v7'
- **KeepEngine**: this option is set by default to on. It means that remote matlab engines are kept/reused between the executions of matlab tasks. Turning this option to off might cause some slowing down and some instability, especially on Windows.
- **Priority**: this option is the priority of jobs created inside the ProActive Scheduler by PAsolve calls. Default to "Normal"
- **ZipInputFiles** and **ZipOutputFiles**: this option tells if PATasks InputFiles/OutputFiles should be zipped before being sent. It is set by default to off. Turning the options on is not recommended due to some instability observed in maltab zip command.
- **FindMatlabScript**: this option sets the script used to find matlab on the remote host. Changing the script means modifying the toolbox internal behavior and is therefore not recommended.
- **MatlabReservationScript**: this option sets the script used to reserve a matlab token (and eventually toolboxes tokens) prior to run matlab code on the remote engine. Again, changing the script means modifying the toolbox internal behavior and is therefore not recommended. Eventually the script itself could be edited to add checks for toolboxes that are not supported by ProActive Scheduler Toolbox : simulink, control system, fixedpoint, images, neural networks, optimization, pde, robust, signal, spline, stats, symbolic, system identification, virtual reality, simulink control design, simulink stateflow, compiler.
- **CustomScript**: In addition to FindMatlabScript and MatlabReservationScript a custom selection script can be added to select only specific resources, such as matlab engine running only on Windows, or from a specific set of machines.
- **ProActiveJars**: libraries used by ProActive Scheduler Toolbox (internal use only).
- **ProActiveConfiguration**: path to ProActive Configuration file (modifying this option requires knowledge of ProActive internals).
- **DisconnectedModeFile**: path to a file used by ProActive Scheduler Toolbox disconnected mode.

Back to Top

Using the PAoptions function

While options can be edited via the PAoptions.ini file, it is still possible to change options on the run via calls to the PAoptions function. The example in this chapter will detail the usage of the option **TransferEnv** which allows the local workspace to be transferred to remote engines.

We will define a function called **testenv** which will read a variable from the parent workspace and will compute the factorial of it. Here is how the **testenv** function is defined:

```
function out=testenv()
a=evalin('caller', 'a');
out = factorial(a);
```

Here is how we set the TransferEnv option dynamically, and execute the call:

```
>> PAoptions('TransferEnv',true);
>> a=5

a =

     5

>> r=PAsolve(@testenv,{})
Job submitted : 11
Awaited (J:11)
>> r.val
[ Thu Jan 13 01:11:35 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:35 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:35 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:35 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:35 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]>> Warning: Element(s) of array '' do not match the current c
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]         definition for class 'PAResult'.  The element(s) hav
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]         to structures.
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]Warning: Could not find appropriate function on path loading
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]/home/fviale/eclipse_workspace/Scheduling_Clean/extensions/ma
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]Warning: Could not find appropriate function on path loading
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]/home/fviale/eclipse_workspace/Scheduling_Clean/extensions/ma
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]Warning: Could not find appropriate function on path loading
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]/home/fviale/eclipse_workspace/Scheduling_Clean/extensions/ma
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]Warning: Could not find appropriate function on path loading
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]/home/fviale/eclipse_workspace/Scheduling_Clean/extensions/ma
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]Warning: Could not find appropriate function on path loading
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]/home/fviale/eclipse_workspace/Scheduling_Clean/extensions/ma
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]Warning: Could not determine the fields for class PAResult by
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]         constructor with no input arguments.  The object res
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]         converted to a structure.  To eliminate the conversi
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]         an object of class PAResult before calling LOAD.
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]Warning: Could not find appropriate function on path loading
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]/home/fviale/eclipse_workspace/Scheduling_Clean/extensions/ma
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]Warning: Could not determine the fields for class PAResult by
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]         constructor with no input arguments.  The object r b
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]         converted to a structure.  To eliminate the conversi
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]         an object of class PAResult before calling LOAD.
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:11:36 CET 2011 ][pendule OUT]


ans =

   120
```

The warnings displayed shows the variables that the remote matlab engine couldn't load properly. These variables are the PAResult variables used in the examples. We ignore these warnings as the PAResult has no meaning in the remote engines. If now, we reset the TransferEnv option to false and try to execute testenv, we will receive the following error:

```
>> PAoptions('TransferEnv',false);
>> r=PAsolve(@testenv,{})
Job submitted : 12
Awaited (J:12)
>> r.val
[ Thu Jan 13 01:20:11 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:20:11 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule ERR] ??? Error using ==> evalin
[ Thu Jan 13 01:20:12 CET 2011 ][pendule ERR]Undefined function or variable 'a'.
[ Thu Jan 13 01:20:12 CET 2011 ][pendule ERR]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule ERR]Error in ==> testenv at 2
[ Thu Jan 13 01:20:12 CET 2011 ][pendule ERR]a=evalin('caller', 'a');
[ Thu Jan 13 01:20:12 CET 2011 ][pendule ERR]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule ERR] ??? Error using ==> save
[ Thu Jan 13 01:20:12 CET 2011 ][pendule ERR]Variable 'out' not found.
[ Thu Jan 13 01:20:12 CET 2011 ][pendule ERR]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:20:12 CET 2011 ][pendule OUT]

??? Error using ==> PAResult.PAwaitFor at 99
Error during remote script execution

Error in ==> PAResult.subsref at 47
                    val{i+1} = PAwaitFor(val{i});
```

Indeed the remote execution of the testenv function couldn't find in its caller workspace the variable "a" !

Back to Top

Debugging

Debugging of the different ProActive Scheduler Toolbox components is done via all of the following:

- The **Debug** option.
- The **CheckMatlabXXX.log** and **ReserveMatlabXXX.log** files.
- The **MatlabTaskXXX.log** files.
- The **AODataspaceRegistry.log** file.

Files with XXX in the above list are created with a complex ID replacing the XXX pattern.

The **CheckMatlab**, **ReserveMatlab** and **MatlabTask** log files are produced on the **worker hosts** *TEMP* directory. The TEMP directory is always */tmp* on linux systems. On Windows it can be known by opening a command line (cmd.exe) and typing echo

%TEMP%:

```
C:\Users\Administrator>echo %TEMP%
C:\Users\Administrator\AppData\Local\Temp\2
```

The **CheckMatlab** log files logs the output of FindMatlab selection scripts which determines where to find matlab according to MatlabConfiguration.xml file (see for more info). Read these files if you want to determine why the task couldn't execute on this worker host. Here is an example of those files:

```
CheckMatlab0a3e377deae6f182_2396e9a2_12d7b483856__8000.log:
Parsing configuration file :/home/fviale/eclipse_workspace/Scheduling_Clean/extensions/matlab/config/worke
Choosing config with version_pref=7.5, versionRej=[7.8, 7.10, 7.6], versionMin=7.5, versionMax=7.11
Version : 7.5
... preferred
Matlab home : /usr/local/matlab2007b
Matlab version : 7.5
Matlab lib directory name : bin/glnxa64
Matlab bin directory  : bin
Matlab command name : matlab
Ptolemy lib dir : /home/fviale/eclipse_workspace/Scheduling_Clean/dist/lib/7.5/bin/glnxa64
```

Similarly the **ReserveMatlab** log files are the output of ReserveMatlab selection scripts which reserve matlab tokens for computations. Here is an example of those files:

```
ReserveMatlab0a3e377deae6f182_2396e9a2_12d7b483856__8000.log:
    Wed Jan 12 18:33:44 CET 2011 : Executing toolbox checking script on pendule
    /usr/local/matlab2007b/bin/matlab
    -nodisplay
    -nojvm
    -nosplash
    -r
    a=1,a=1,if a,fid = fopen('/tmp/0a3e377deae6f182_2396e9a2_12d7b483856__8000/matlabTest1.lock','w'),fcl
    command executed
    OK
    Wed Jan 12 18:34:00 CET 2011 : Executing toolbox checking script on pendule
    /usr/local/matlab2007b/bin/matlab
    -nodisplay
    -nojvm
    -nosplash
    -r
    a=1,a=1,if a,fid = fopen('/tmp/0a3e377deae6f182_2396e9a2_12d7b483856__8000/matlabTest1.lock','w'),fcl
    command executed
    OK
```

The **MatlabTask** log files are not created by default. They are the verbose output of the tasks submitted via PAsolve and are activated only if the PAoption Debug is set to on. Here is an example of those files:

```
MatlabTask0a3e377deae6f182_2396e9a2_12d7b483856__8000.log:
    [Thu Jan 13 01:56:33 CET 2011 pendule MatlabTask] Checking Processes...
    [Thu Jan 13 01:56:33 CET 2011 pendule MatlabTask] Executing the task, try 1
    [Thu Jan 13 01:56:33 CET 2011 pendule MatlabTask] Initializing : 2396e9a2-12d7b483856--7651--0a3e377de
    [Thu Jan 13 01:56:33 CET 2011 pendule MatlabTask] Executing
    [ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.put(test, 1)
    [ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]ptmatlabCreateDoubleMatrix(test) 1 x 1
    [ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]ptmatlabEngPutArray(test) 1 x 1
    [ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]ptmatlabDestroy(test)
    [ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]ptmatlabEngGetArray(test) 1 x 1
    [ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]ptmatlabGetClassName() = double
```

```
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]ptmatlabGetDimensions() = 1 x 1
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]ptmatlabIsComplex() = 0
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]ptmatlabGetDoubleMatrix() 1 x 1
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]ptmatlabDestroy(test)
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.get(test) = 1
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("clear test")
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("clear all;")
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("cd('/tmp/PA_JVM1485855611/PA_JVM
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]Unzipping source files
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("restoredefaultpath;")
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("addpath('/tmp/PA_JVM1485855611/
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("unzip('/tmp/PA_JVM1485855611/PA_
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]Contents of /tmp/PA_JVM1485855611/PA_JVM1485855611_GCMNod
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]/tmp/PA_JVM1485855611/PA_JVM1485855611_GCMNode-0/13/2396e
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]/tmp/PA_JVM1485855611/PA_JVM1485855611_GCMNode-0/13/2396e
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]/tmp/PA_JVM1485855611/PA_JVM1485855611_GCMNode-0/13/2396e
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]Loading Input Variable file /tmp/PA_JVM1485855611/PA_JVM
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("load('/tmp/PA_JVM1485855611/PA_J
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("who")
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]>>
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]Your variables are:
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]RESTOREDEFAULTPATH_EXECUTED
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]in1
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]Feeding input:
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]i=0;
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("i=0;")
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]Executing Matlab command:
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]out = factorial(in1);
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("out = factorial(in1);
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]")
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]Matlab command completed successfully
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]Receiving output:
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]Saving Output Variable file /tmp/PA_JVM1485855611/PA_JVM
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("save('/tmp/PA_JVM1485855611/PA_J
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]true
[Thu Jan 13 01:56:33 CET 2011 pendule MatlabTask] Received result
[Thu Jan 13 01:56:33 CET 2011 pendule MatlabTask] Packing memory in Matlabengine
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("rmpath('/tmp/PA_JVM1485855611/PA
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("v=fopen('all');for i=1:length(v
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("cd('/tmp');")
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("clear all;")
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]
[ Thu Jan 13 01:56:33 CET 2011 ][pendule OUT]matlabEngine.evalString("pack;")
[Thu Jan 13 01:56:34 CET 2011 pendule MatlabTask] Task completed successfully
[ Thu Jan 13 01:56:34 CET 2011 ][pendule OUT]
```

```
[Thu Jan 13 01:56:34 CET 2011 pendule MatlabTask] Performing after task actions
[Thu Jan 13 01:56:34 CET 2011 pendule MatlabTask] Closing output
```

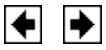
Finally the **AODataspaceRegistry.log** file is created on the **client host** and is logging everything concerning ProActive DataSpaces. Here is an example of those files:

```
Looking up or creating dataspaces for :/home/fviale/matlab/Model_22_06_2009
Creating new dataspaces
Input dataspace created at url : paprmi://pendule.inria.fr:1099/MatlabInputSpace_e930b2ec?proactive_vfs_p
Output dataspace created at url : paprmi://pendule.inria.fr:1099/MatlabOutputSpace_e930b2ec?proactive_vfs
Looking up or creating dataspaces for :/home/fviale/matlab/Model_22_06_2009
Creating new dataspaces
Input dataspace created at url : paprmi://pendule.inria.fr:1099/MatlabInputSpace_e930b2ec?proactive_vfs_p
Output dataspace created at url : paprmi://pendule.inria.fr:1099/MatlabOutputSpace_e930b2ec?proactive_vfs
Looking up or creating dataspaces for :/home/fviale/matlab/Model_22_06_2009
Reusing existing dataspaces
Looking up or creating dataspaces for :/home/fviale/matlab/Model_22_06_2009
Creating new dataspaces
Input dataspace created at url : paprmi://pendule.inria.fr:1099/MatlabInputSpace_e930b2ec?proactive_vfs_p
Output dataspace created at url : paprmi://pendule.inria.fr:1099/MatlabOutputSpace_e930b2ec?proactive_vfs
```

Back to Top

Disconnected mode

Using ProActive Scheduler Toolbox Simulink GUI



Disconnected mode

On this page…

Introduction

Exiting Matlab before a job completion

Reconnecting and Retrieving previous job results

Introduction

In the previous chapter we saw how we can run complex matlab code remotely, and wait for the results produced. Sometimes, when tasks are really long and matlab licences scarce, it can be very convenient to close the local matlab session to avoid consuming a token or local processing resources. In this chapter, we will learn how to use the **Disconnected Mode** feature of **ProActive Scheduler Toolbox**.

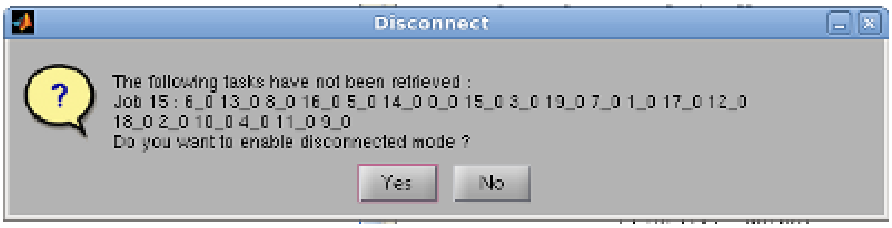
Back to Top

Exiting Matlab before a job completion

In this example we will run a long list of factorial tasks and will close the matlab session:

```
>> res=PAsolve(@factorial,1,2,3,4,5,6,7,8,9,10,1,2,3,4,5,6,7,8,9,10)
Job submitted : 15
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
Awaited (J:15)
>> exit
```

At this point a popup window will appear, asking if we want to enable disconnected mode, with the list of jobs unfinished.



We can focus on other works while the job is computed, but with a limitation: we cannot turn off the computer as the local DataSpace is still used by the job to produce input and output files. An alternative exists though, we can use a Custom Dataspace as explained in chapter The PAoptions.ini file to even allow turning off the computer !

Back to Top

Reconnecting and Retrieving previous job results

When we restart the matlab session, we will need to reconnect to ProActive Scheduler in order to receive the job results. The **PAconnect** function will display the list of jobs that were unfinished at last session. We will then use the function **PAgetResults** to receive the results :

```
>> PAconnect('rmi://pendule:1099/')
Connection successful, please enter login/password
Login succesful
Reconnecting to existing dataspace handler, please wait...
The following jobs were uncomplete before last matlab shutdown :  15

ans =

    '15'

>> res=PAgetResults(15)
Retrieving results of job 15
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:20 CET 2011 ][pendule OUT]


res =

     1

[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]
```

```
[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]
[ Thu Jan 13 02:21:14 CET 2011 ][pendule OUT]


res =

     2
...
```

The previous output has been stripped for clarity. Sometimes it can be complicated to remember to which computation is associated the JobID, but it is possible to read the **jobid** attribute from the **PAresult** object right after submitting via PAsolve (non-blocking call) :

```
>> res2=PAsolve(@factorial,1,2,3,4,5)
Job submitted : 16
Awaited (J:16)
Awaited (J:16)
Awaited (J:16)
Awaited (J:16)
Awaited (J:16)

>> res2.jobid

ans =

16


ans =

16


ans =

16


ans =

16


ans =

16
```

This allows for example to store the job id into a mat file and automate the retrieval of results avec reconnection.

Back to Top

Running Matlab functions remotely Configuring PAsolve behavior and Debugging

Using ProActive Scheduler Toolbox Simulink GUI

On this page…

Introduction

Starting the GUI

Defining model and parameters

Running the model

Viewing results

Using Disconnected Mode

Introduction

The **ProActive Scheduler Toolbox Simulink GUI** is a simple GUI which allows to run a Simulink model on a remote engine. It allows to do batch simulations through the use of Matlab script initialization parameters. Thus, the same model can be run multiple times with parameters. The GUI is NOT an interface similar in its complexity to the Simulink interface, it doesn't allow real time display of the simulation. But results from the simulations can be displayed graphically.

In order to display the results the model MUST save remotely output signals to mat files via the **To File** block. Usages in the model of **Scope** or **To workspace** should be avoided as it would slow down the remote simulation and consume memory.

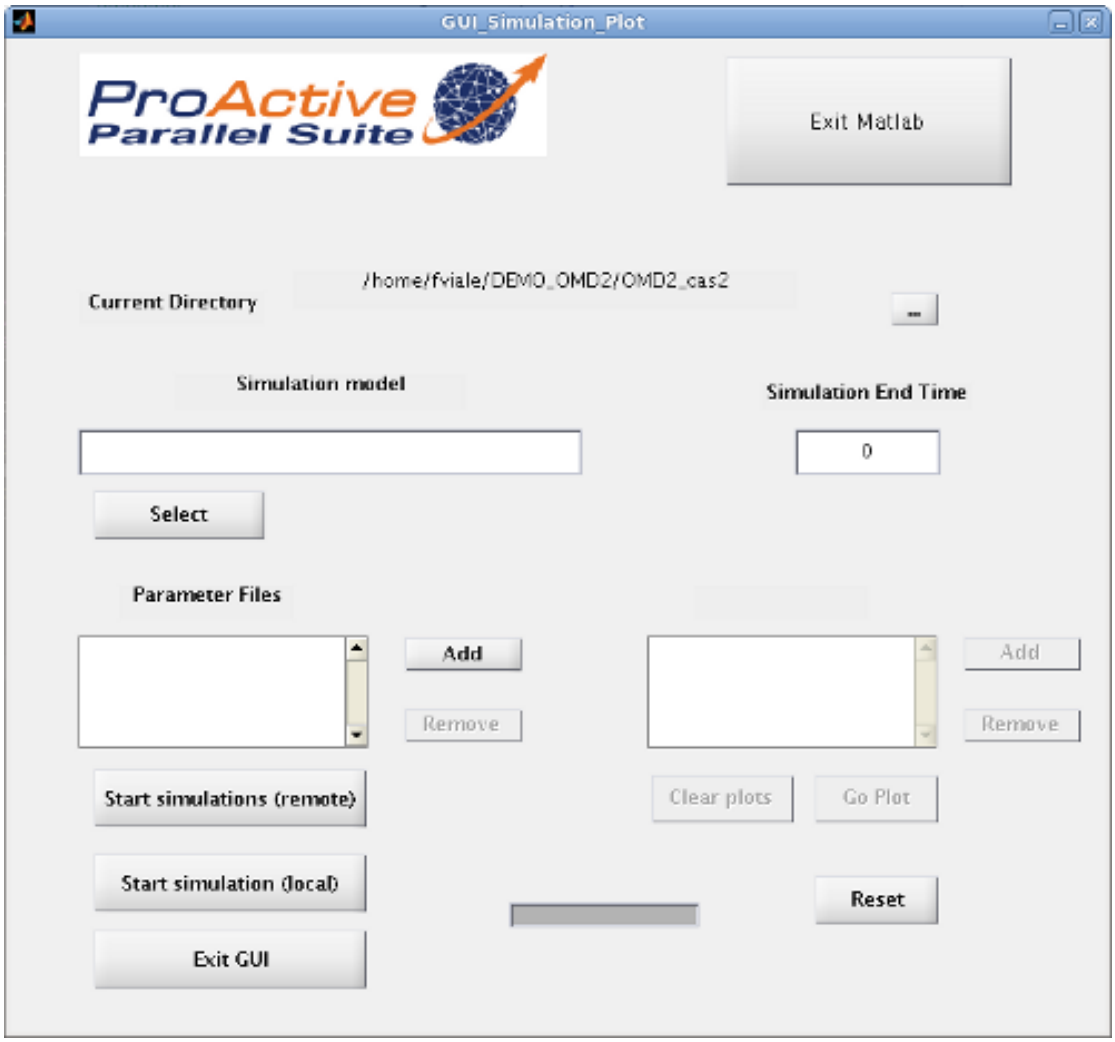
Back to Top

Starting the GUI

The **ProActive Scheduler Toolbox Simulink GUI** is located at SCHEDULING/extensions/matlab/toolbox/GUI_Simulink , before opening the GUI, make sure it's available in matlab path. Opening the GUI can be done either by running the function **GUI_Simulation_Plot**, or via Matlab start button at **Start > Toolboxes > ProActive Scheduler > Scheduler Simulink GUI :**

```
>> addpath('/home/fviale/eclipse_workspace/Scheduling_Clean/extensions/matlab/toolbox/GUI_Simulink/');
>> GUI_Simulation_Plot
```

At startup, a popup window appears asking if we want to load an existing model, we answer No for now. Here is a view of the GUI:



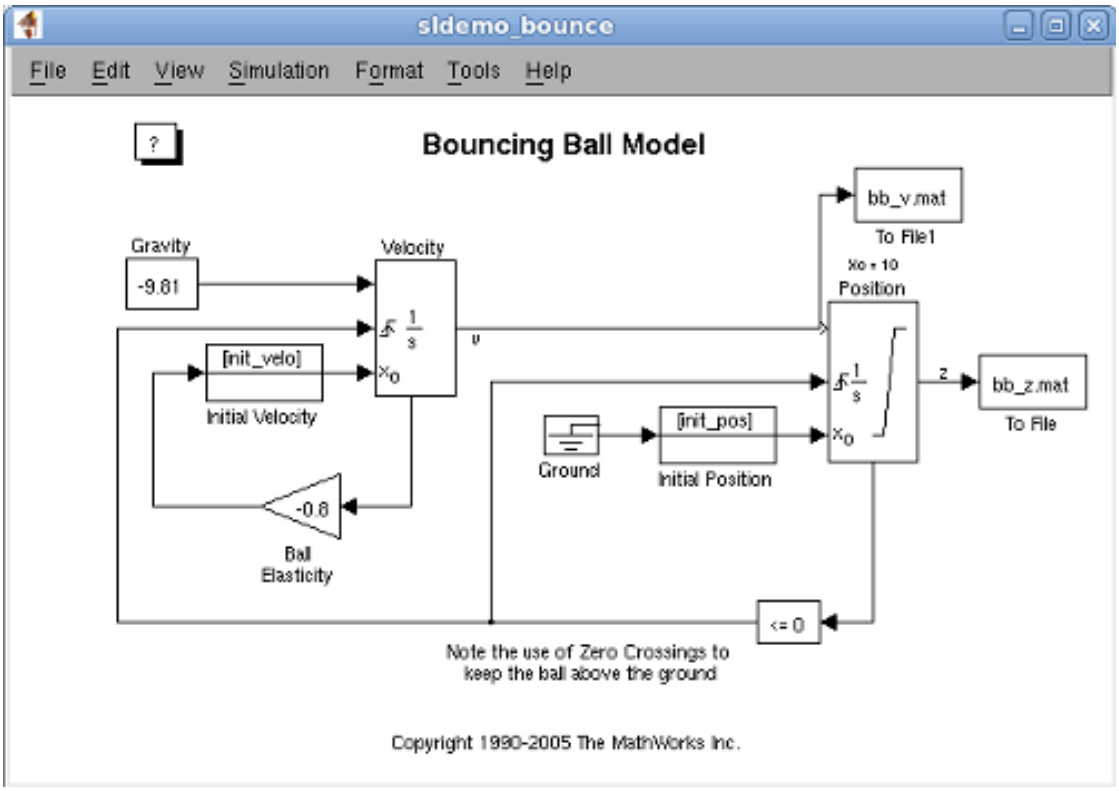
Back to Top

Defining model and parameters

We will now create a model to use with the GUI and a set of parameters. We will reuse the Simulink Demo **Bouncing Ball Model** for this purpose. We edit the bouncing ball demo to fit to our use, doing the following:

- We remove the two **Scope** outputs present in the demo and replace it with **To file** blocks.
- We modify the **Initial Velocity** and **Initial Position** blocks by changing their initial value from a constant to a user defined variable.

Here is how the model looks like (available at SCHEDULING/extensions/matlab/toolbox/GUI_Simulink/demo):



We will then write 3 matlab scripts which will assign the values of the **init_velo** and **init_pos** variables:

```
Param1.m:
init_velo=15;
init_pos=10;

Param2.m:
init_velo=10;
init_pos=30;

Param3.m:
init_velo=10;
init_pos=30;
```

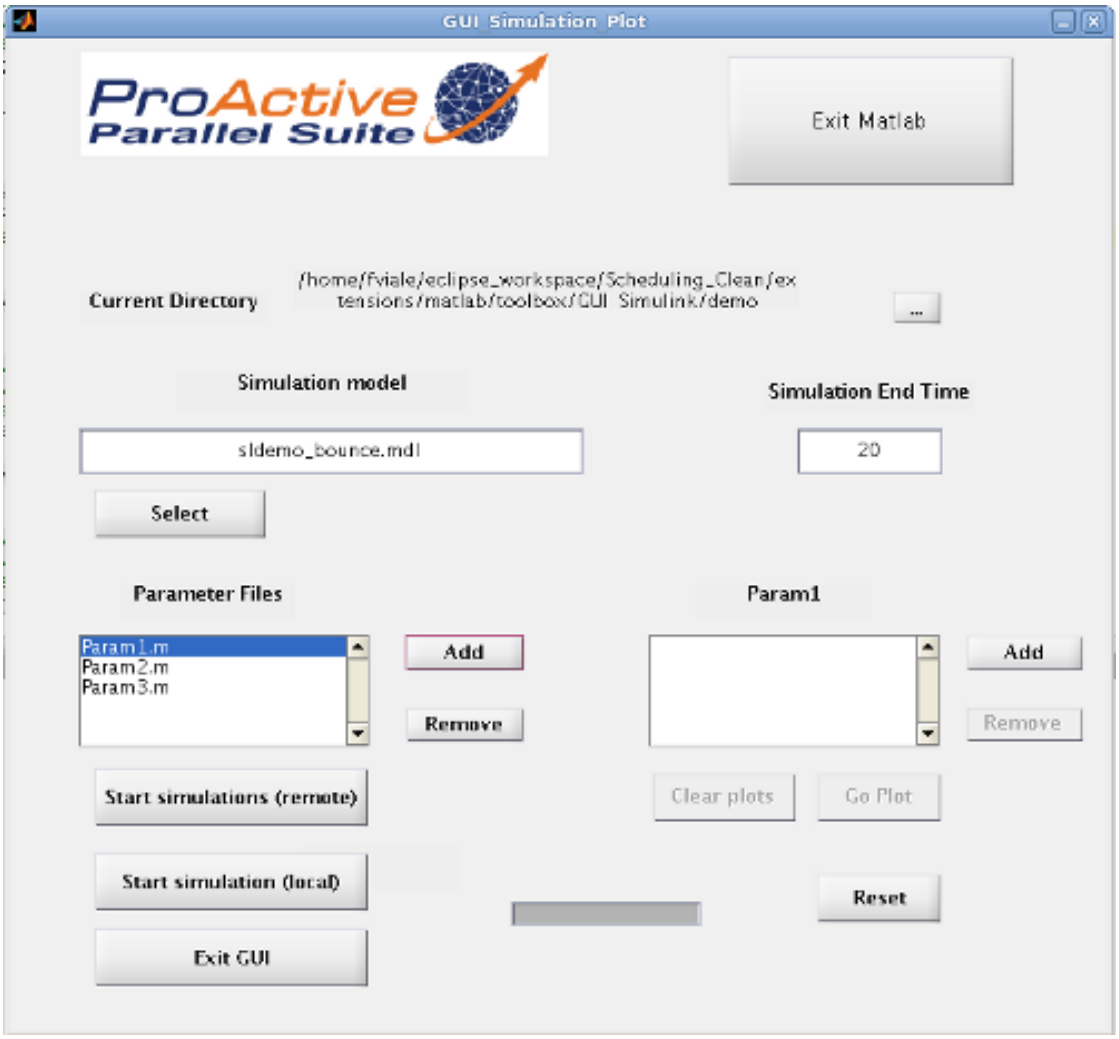
Back to Top

Running the model

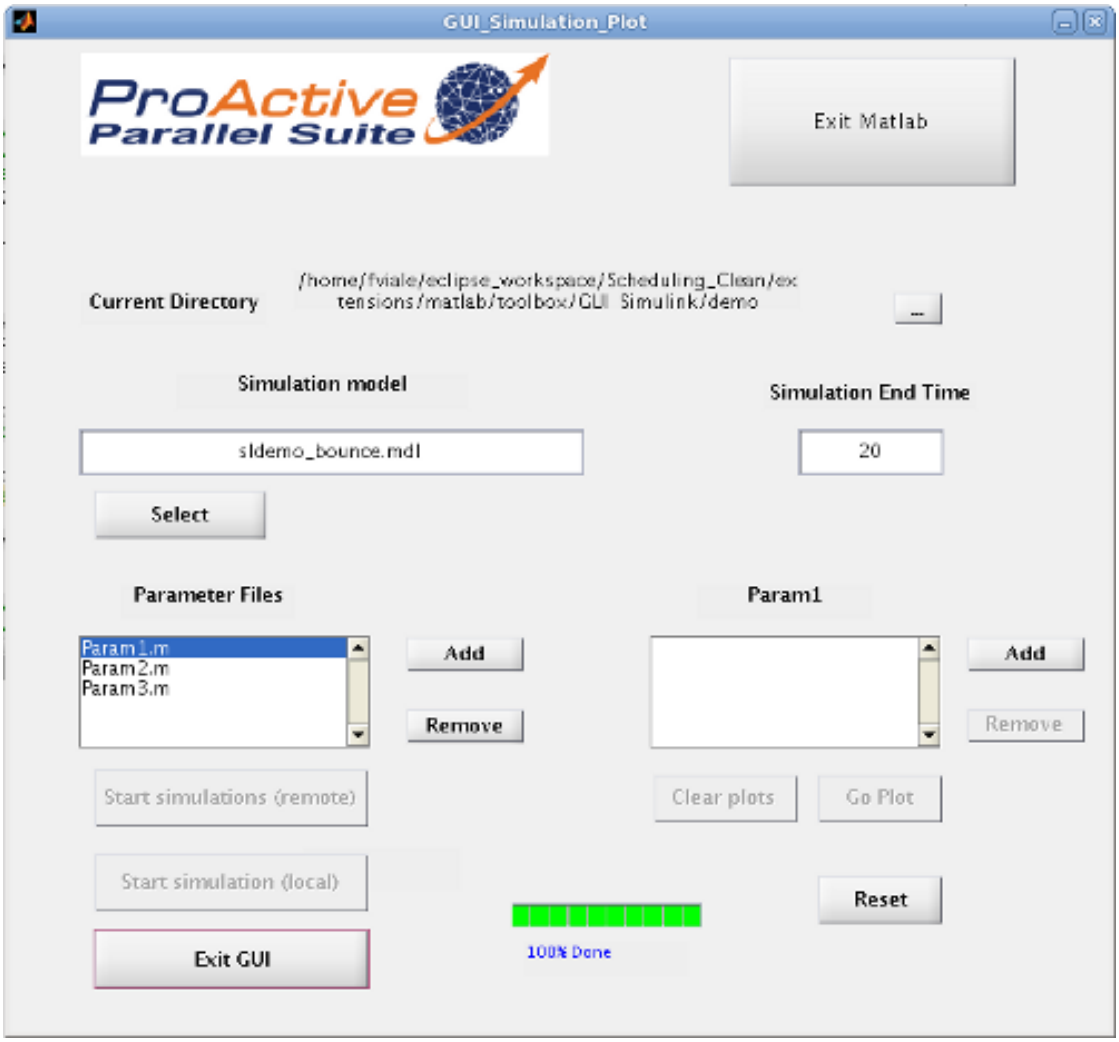
We are now ready to run the model. We must make sure beforehand that a connection to the scheduler is established. We will open the GUI and do the following actions:

1. We change the **Current Directory** to the directory where the model and parameter files are (they must be in the same directory, here SCHEDULING/extensions/matlab/toolbox/GUI_Simulink/demo).
2. We select the model in the **Simulation Model** field (here sldemo_bounce.mdl).
3. We set the **Simulation End Time**, it will be the same for all batches run (here 20).
4. We add the 3 ParamX.m files that we created in the **Parameter Files** list.
5. Finally we press the **Start Simulations (remote)** button.

Here is how the GUI looks like before pressing start:



Here is how the GUI looks like after all simulations are executed:



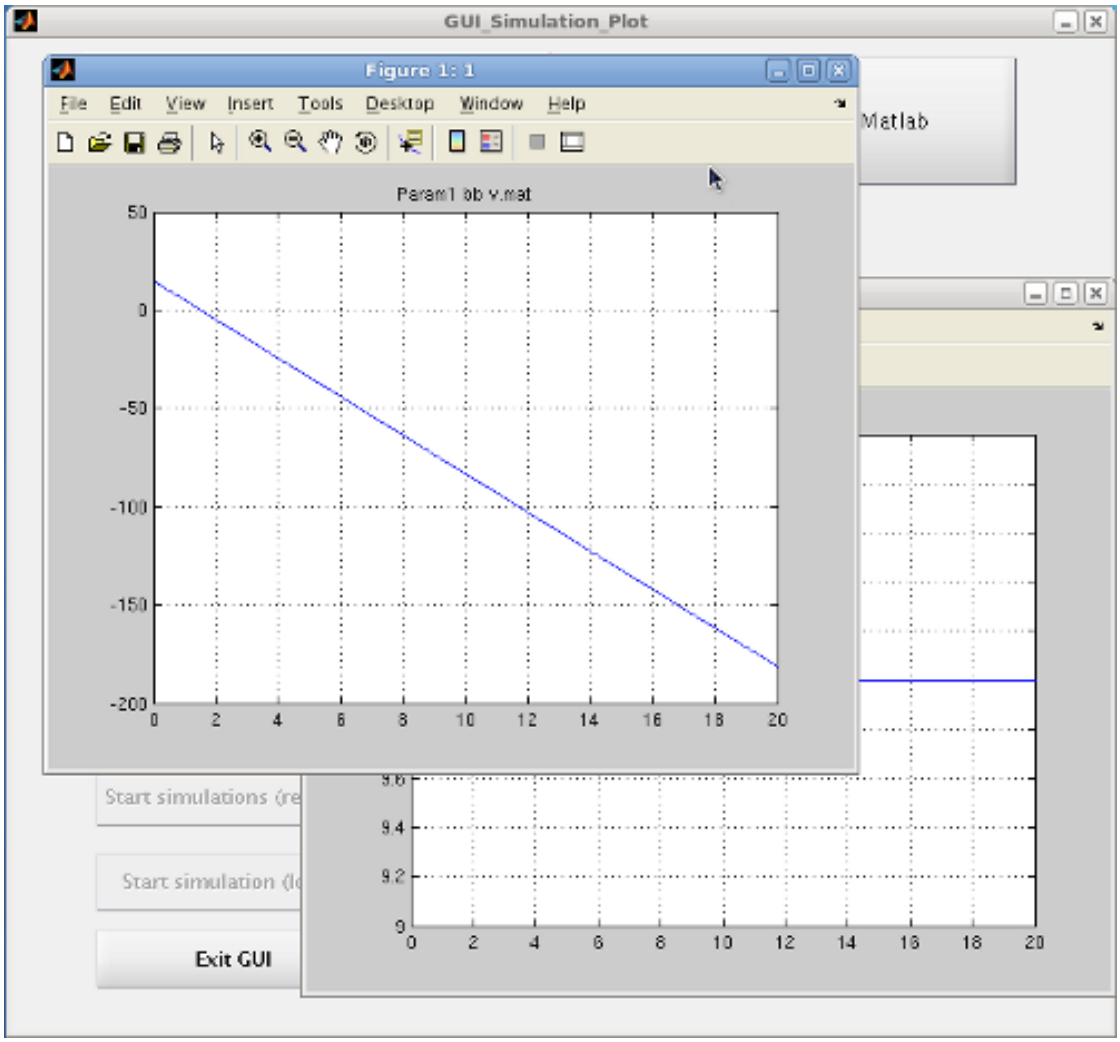
Back to Top

Viewing results

Now that all simulations are executed, we want to visualize the results. This is done via the list box on the right side of the GUI.

1. We first need to select on the left list (**Parameter Files**) the parameter file from which we want to display results. Here we choose **Param1**.
2. We will then press the **Add** button to add signals to be visualized. Here we will choose the two output files **bb_v.mat** and **bb_z.mat**.
3. Finally we press the **Go Plot** button which will pop up two graphical plot windows displaying the results.

Here is the results of the simulation with Param1:



We can repeat the previous steps for the other parameter files Param2 nad Param3.

Back to Top

Using Disconnected Mode

It is possible to use the GUI in combination with **ProActive Scheduler Toolbox Disconnected Mode**. In order to do that:

1. Submit a simulation.
2. Hit the button Exit Matlab. Accept the disconnected mode.
3. At next Matlab Startup reconnect to the scheduler and reopen the GUI.
4. A popup window will appear saying that an existing running model has been found. Accept to reload it.
5. The simulation will be updated to its current state and if the simulation is finished results will be accessible.

Back to Top

Configuring PAsolve behavior and Debugging