

スーパーコンピュータでMATLAB を動かすハンズオン実習

2025/07/30

MathWorks Japan

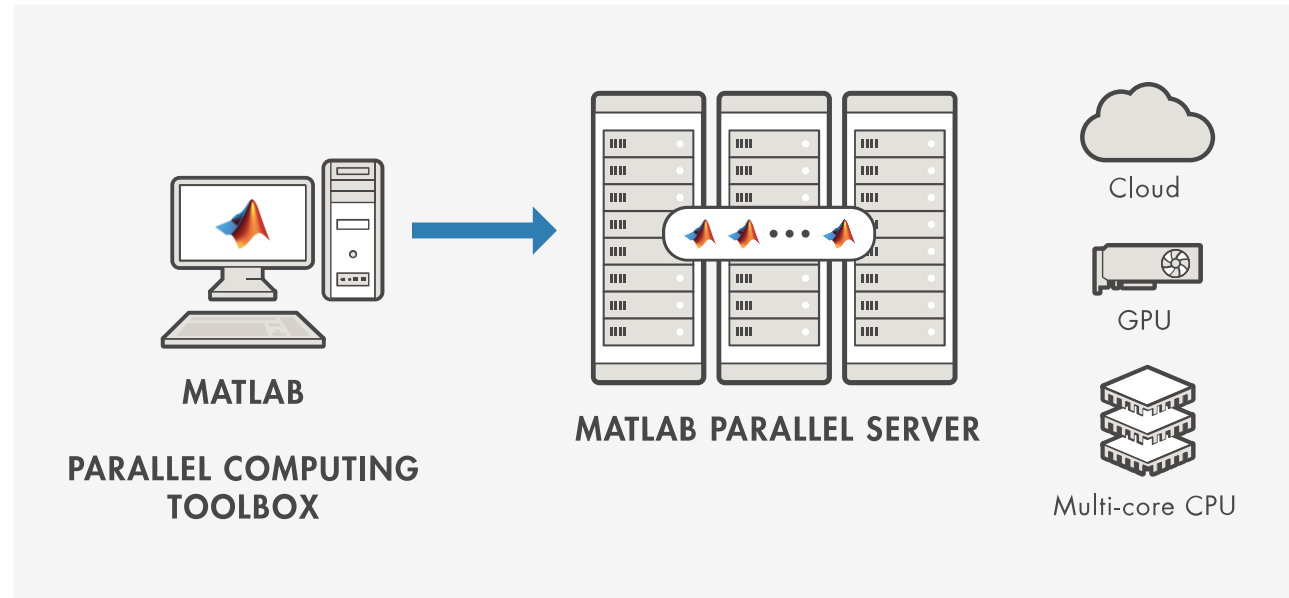
アプリケーションエンジニアリング部 テクニカルコンピューティング

齊藤 甲次郎 (ksaito@mathworks.com)

アジェンダ

- スーパーコンピュータでのMATLAB 実演
 - MATLAB の並列処理
 - スーパーコンピュータでのMATLAB 実演
 - Open OnDemand を使用したMATLAB の利用
 - インタラクティブな並列処理の実演
 - オフロードで投げるバッチ処理の実演
 - 実験マネージャアプリを使った並列処理
 - Fortran、CのプログラムからMATLAB を呼び出す実演

MATLAB の並列処理



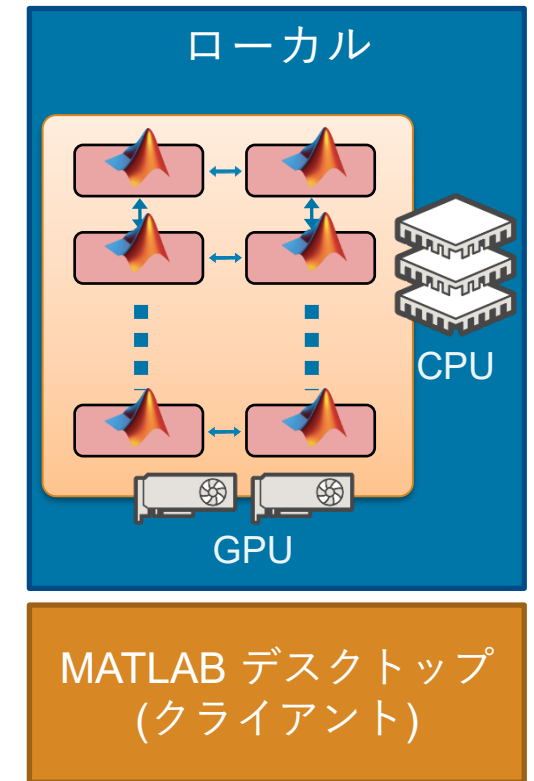
Parallel Computing Toolbox

1台のマシンでの MATLAB 関数及び Simulink モデルの並列実行

- ローカルでの MATLAB & Simulink 製品ファミリーと連携した並列処理による演算の高速化
 - 並列 for ループ
 - 並列アルゴリズムの使用
 - GPU 演算
 - バッチジョブの並列実行
- ジョブおよびタスクの制御
- ビッグデータ解析の分割処理
 - メモリに収まらないデータの扱い
 - スパース分散行列の作成



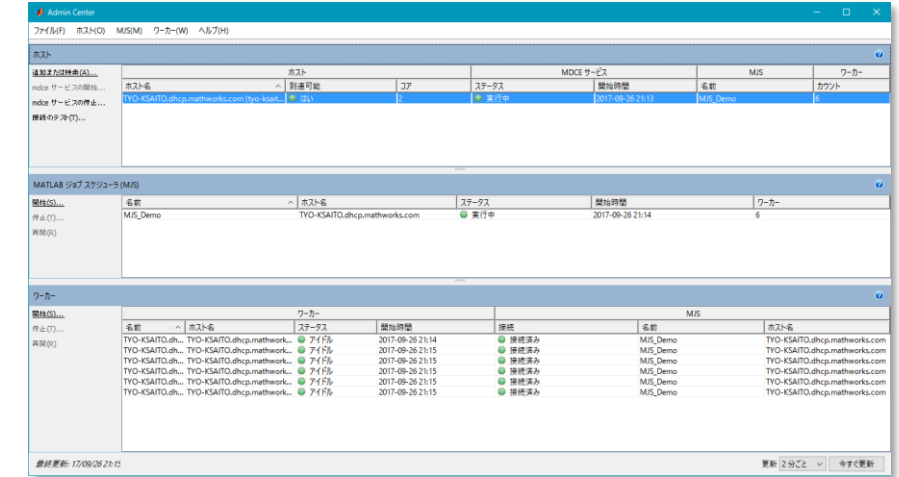
デスクトップコンピュータ



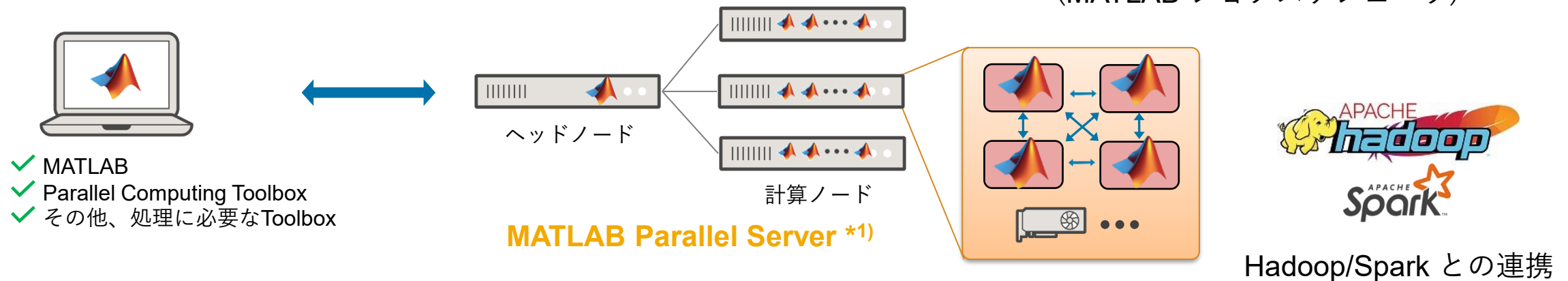
MATLAB Parallel Server

クラスター環境を利用した MATLAB 関数及び Simulink モデルの並列実行

- クラスタマシンのマルチCPU・GPUでの並列処理
- 簡易なスケジューラ(MATLAB ジョブスケジューラ)
- サードパーティスケジューラとの連携
- Hadoop®・Spark®との連携によるビッグデータ処理
- クライアントの製品構成に依存しない計算環境を提供

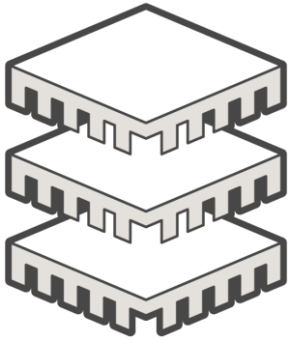
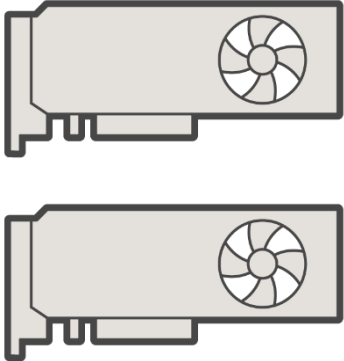


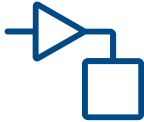






GUI による簡単な操作
(MATLAB ジョブスケジューラ)



*1) R2018b までの名称は MATLAB Distributed Computing Server (MDCS)

並列処理の種類

CPU	GPU	マシンの台数
		
<p data-bbox="264 833 738 879">主なアプリケーション</p> <div data-bbox="183 931 779 1245"><p data-bbox="188 1110 377 1159">機械学習</p><p data-bbox="336 1205 713 1245">シミュレーション</p><p data-bbox="631 1090 779 1139">最適化</p></div>	<p data-bbox="1014 833 1488 879">主なアプリケーション</p> <div data-bbox="952 925 1589 1216"><p data-bbox="958 1053 1146 1102">画像処理</p><p data-bbox="1156 1173 1589 1216">ディープラーニング</p></div>	

並列処理の関数

1. インタラクティブジョブ
 - parfor: 汎用的な並列処理
 - spmd: データを分割して同じ処理を実行 (single process multiple data)
 - parsim: 並列シミュレーション
 - parfeval: 非同期の並列処理

```
for n = 1:numel(partitions)-1
    f(n) = parfeval(@parameterSweep,1,partitions(n),partitions(n+1),sigma,rho,beta,Q);
end
```

parfor の例

```
n = 200;
A = 500;
a = zeros(n);
parfor i = 1:n
    a(i) = max(abs(eig(rand(A)))));
end
```

spmd の例 (円周率の計算)

```
spmd
    a = (labindex - 1)/numlabs;
    b = labindex/numlabs;
    fprintf('Subinterval: [%-4g, %-4g]\n', a, b);
end

spmd
    myIntegral = integral(@pctdemo_aux_quadpi, a, b);
    fprintf('Subinterval: [%-4g, %-4g] Integral: %4g\n', ...
        a, b, myIntegral);
end

% 結果の合計
spmd
    piApprox = gplus(myIntegral);
end
```

バッチの例

並列処理

2. オフロードジョブ

- バッチ:

クラスターでMATLAB スクリプトを実行

- 独立ジョブ:

クラスターの1ワーカーで実行するジョブ

- 通信ジョブ:

クラスターの複数ワーカーで実行するジョブ

```
% クラスターの作成
c = parcluster();
% バッチ処理の実行
j = batch(c,@rand,1,{10,10},
'CaptureDiary', ...
true, 'CurrentFolder', '.');
% ジョブの終了待ち
wait(j)
% ジョブ結果の回収
out = fetchOutputs(j);
% ジョブの消去
delete(j)
```

通信ジョブの例

```
% クラスターの作成
c = parcluster;
% ジョブの作成
j = createCommunicatingJob(c,'Type',
'pool');
% タスクの作成
createTask(j, @myFunction, 1, {100});
% ジョブの投入
submit(j);
% ジョブの終了待ち
wait(j)
% ジョブ結果の回収
out = fetchOutputs(j)
% ジョブの消去
delete(j)

%% カスタム関数
function result = myFunction(N)
    result = 0;
    parfor ii=1:N
        result = result +
max(eig(rand(ii)));
    end
end
```

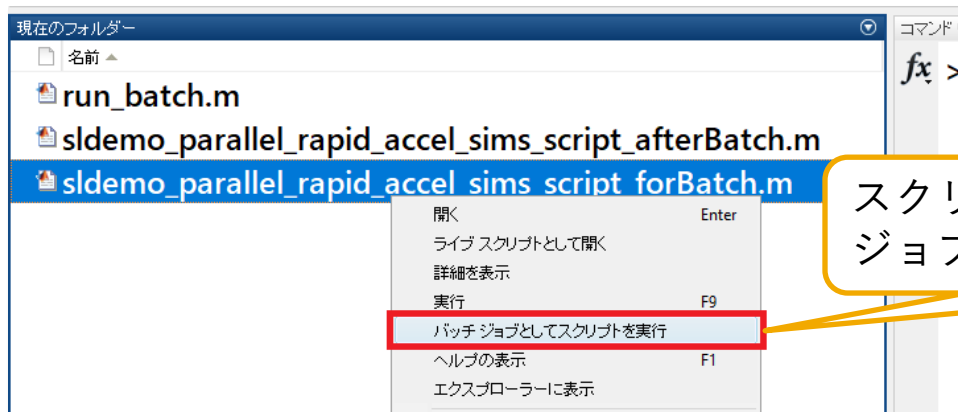
独立ジョブの例

```
% クラスターの作成
c = parcluster
% ジョブの作成
j = createJob(c);
% タスクの作成
for ii = 1:10
    createTask(j,@rand,1,{10});
end
% ジョブの投入
submit(j);
% ジョブの終了待ち
wait(j);
% ジョブ結果の回収
out = fetchOutputs(j);
% ジョブの消去
delete(j)
```


オフロード処理

サーバー上の 1 ワーカーで実行

- バッチスクリプトを 1 ワーカーだけで実行
 - MATLAB の GUI から



スクリプトを右クリックして「バッチ
ジョブとしてスクリプトを実行」



- MATLAB のコマンドから .m を抜いたスクリプト名

```
batch('SCRIPTNAME')
```

例

```
batch('stateflow_MDCS_parsim_forBatch')
```

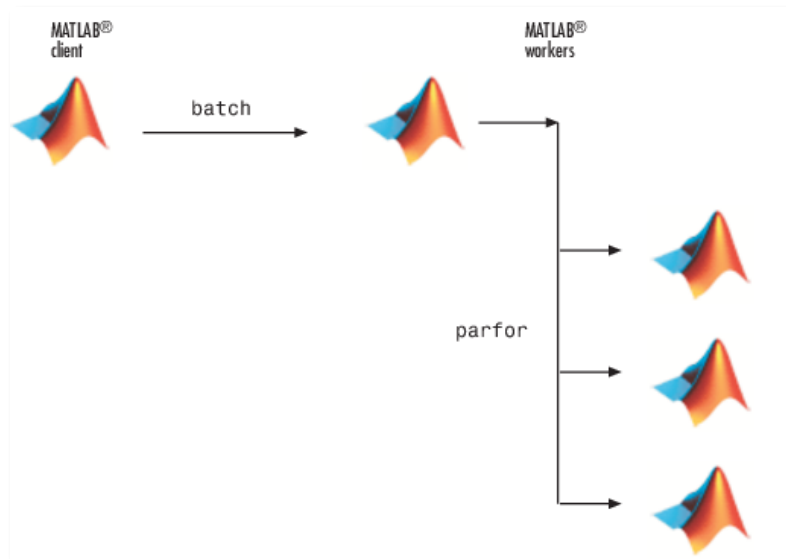
オフロード処理

サーバー上の複数ワーカーで実行

- バッチを使ってspmd やparfor、 parsim を含むスクリプトを複数ワーカーで処理
 - MATLAB のコマンドから

例：MATLAB Parallel Server の 32 ワーカーのライセンスを持っている場合

```
batch('stateflow_MDCS_parsim_forBatch', 'Pool', 31)
```



最大ワーカー数 - 1

プロセスベースとスレッドベースの並列プール

プロセスベースの並列プール (従来どおり)	スレッドベースの並列プール (R2020a 以降)
parpool('Processes')	parpool('Threads')
<p>ワーカー</p> <p>プロセス</p> <p>メモリ</p> <p>データ転送</p> <p>スレッド</p> <p>それぞれのメモリが独立</p>	<p>ワーカー</p> <p>プロセス</p> <p>スレッド間的高速な通信</p> <p>メモリ</p> <p>メモリを共有</p> <p>スレッド</p>
特徴 <ul style="list-style-type: none"> 並列言語を完全にサポート 以前のリリースとの下位互換性 クラッシュ時のより高いロバスト性 外部ライブラリをスレッドセーフにする必要なし 複数台にまたぐ並列処理にも拡張可能 	特徴 <ul style="list-style-type: none"> 並列コードがスレッドベースの環境でサポートされている場合は有効 データ転送量が多い(>100 MB) 場合は有効 メモリ使用量の削減、スケジューリングの高速化、データ転送の低減 1台のみの並列処理をサポート

Toolbox の関数での並列処理

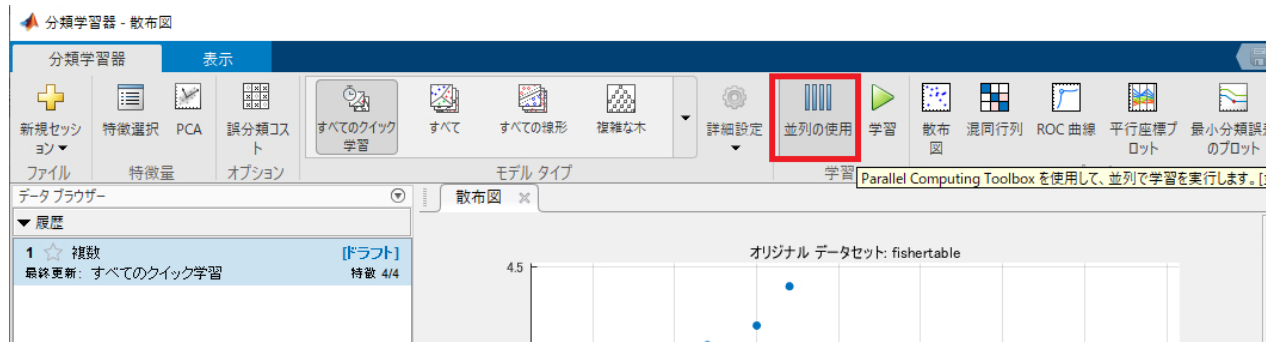
関数のオプションで指定

`optimoptions(..., 'UseParallel', true)` **Optimization Toolbox**

`trainingOptions(..., 'ExecutionEnvironment', 'parallel-auto')` **Deep Learning Toolbox**

既定のクラスタープロファイルの複数GPU、またはCPUを使用

アプリのオプションで変更



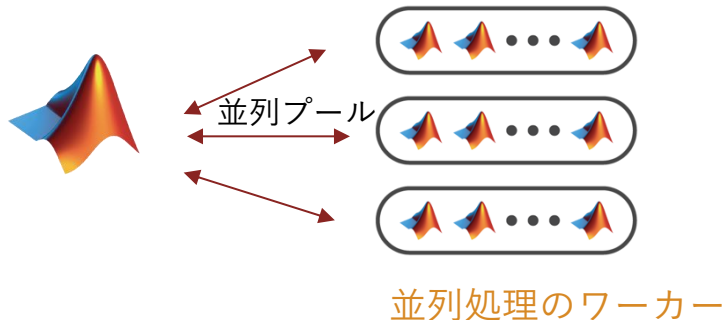
Simulink の並列シミュレーションコマンドで

```
model = 'vdp';  
in = Simulink.SimulationInput(model);  
out = parsim(in)  
%out = batchsim(in)
```

MATLAB のクラスターサーバーへのジョブの投入の方法

インタラクティブジョブ

```
n = 200;  
A = 500;  
a = zeros(1,n);  
parfor i = 1:n  
    a(i) = max(abs(eig(rand(A)))));  
end
```

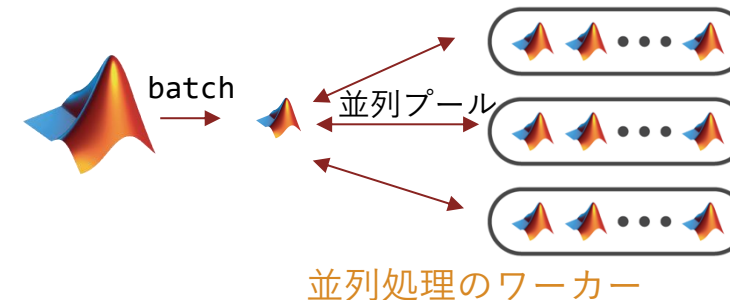


オフロードジョブ

```
n = 200;  
A = 500;  
a = zeros(1,n);  
parfor i = 1:n  
    a(i) = max(abs(eig(rand(A)))));  
end
```

myparfor.m

```
job = batch('myparfor', 'Pool', 3);  
wait(job)  
out = fetchOutputs(job);  
out = out{1};
```



GPU での並列処理

gpuArray の使用

C言語 + CUDA
数百～数千行

```
#include "cufft.h"
#include "cuda_runtime.h"

void pack_r2c(cufftComplex *output_float,
              double *input_re,
              int N)
{
    /* Allocate array */
    cudaMalloc((void **)&rhs_complex_d, sizeof(cufftComplex)*N*M);

    /* Copy input array to device */
    cudaMemcpy(&rhs_complex_d, input_single,
               sizeof(cufftComplex)*N*M, cudaMemcpyHostToDevice);
}
```

MATLAB
での記述

% GPU上で配列を作成

```
GX = gpuArray(X);
```

% GPU上での演算

```
GY = fft2(GX);
```

% ローカルワークスペースへ転送

```
Y = gather(GY);
```

関数のオプションで GPU 配列を作成

```
ones(10,1,'gpuArray');
```

GPU 配列をサポートする Toolbox

ツールボックス名	gpuArray をサポートする関数のリスト
MATLAB	gpuArray をサポートする関数
Statistics and Machine Learning Toolbox™	gpuArray をサポートする関数 (Statistics and Machine Learning Toolbox)
Image Processing Toolbox™	gpuArray をサポートする関数 (Image Processing Toolbox)
Deep Learning Toolbox™	gpuArray をサポートする関数 (Deep Learning Toolbox) *(GPU を使用した深層学習も参照)
Computer Vision Toolbox™	gpuArray をサポートする関数 (Computer Vision Toolbox)
Communications Toolbox™	gpuArray をサポートする関数 (Communications Toolbox)
Signal Processing Toolbox™	gpuArray をサポートする関数 (Signal Processing Toolbox)
Audio Toolbox™	gpuArray をサポートする関数 (Audio Toolbox)
Wavelet Toolbox™	gpuArray をサポートする関数 (Wavelet Toolbox)
Curve Fitting Toolbox™	gpuArray をサポートする関数 (Curve Fitting Toolbox)

ディープラーニングは自動的に GPU で実行

- `trainNetwork` (Deep Learning Toolbox)
- `predict` (Deep Learning Toolbox)
- `predictAndUpdateState` (Deep Learning Toolbox)
- `classify` (Deep Learning Toolbox)
- `classifyAndUpdateState` (Deep Learning Toolbox)
- `activations` (Deep Learning Toolbox)

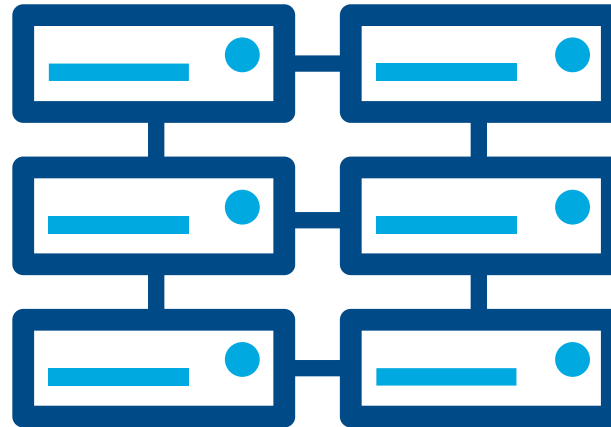
関数のオプションでも指定可能

```
trainingOptions(..., 'ExecutionEnvironment', 'gpu')
```

1台で複数 GPU を使用

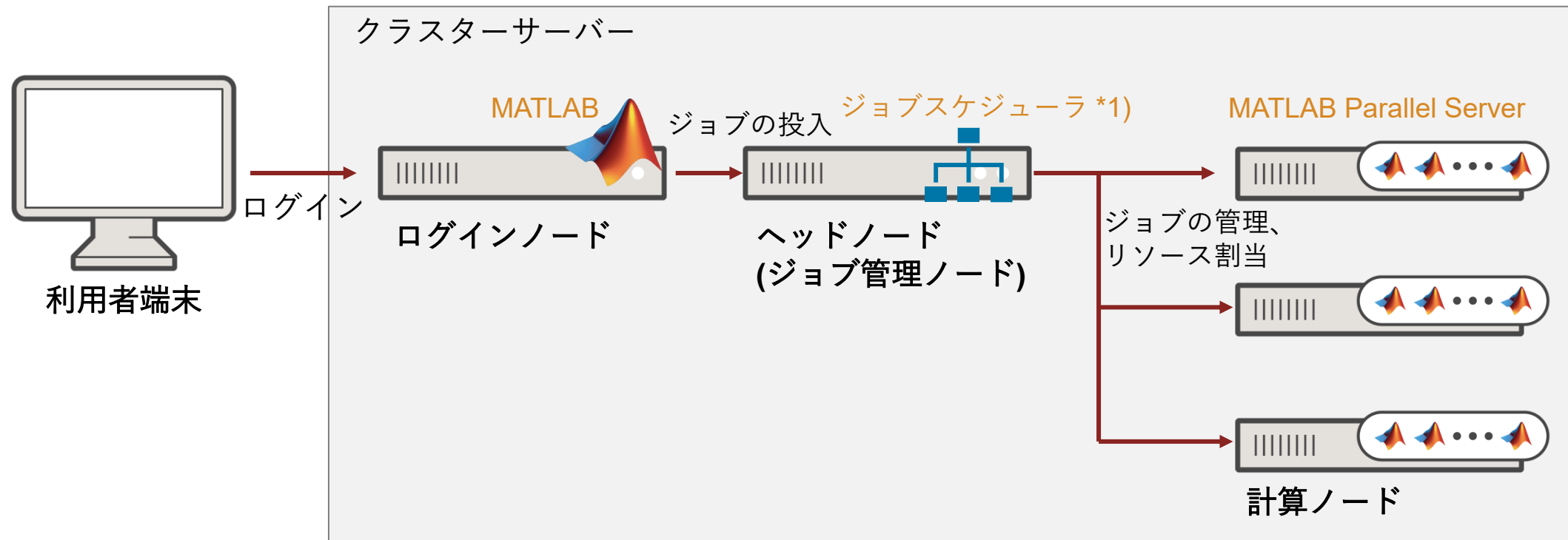
```
trainingOptions(..., 'ExecutionEnvironment', 'multi-gpu')
```

スーパーコンピューターでのMATLAB 実演



クラスターサーバーでのMATLAB の並列処理

クラスターサーバーではジョブの管理やリソース割当をするために、**ジョブスケジューラ**にジョブを投入する必要があります



*1) Miyabi のジョブスケジューラはPBS Professional

一般的なクラスターサーバーへのジョブの投入の方法

- ジョブ投入のためのスクリプトを書きます

ジョブ投入のためのシェルスクリプト (myscript.sh) を作成

```
#!/bin/bash
#PBS -q lecture-c
#PBS -l select=1
#PBS -W group_list=group1
#PBS -j oe

cd ${PBS_O_WORKDIR}
./a.out
```

使用できるキューは **lecture-c** または **tutorial-c** (本講習会中のみ) です

- ジョブを投入します

ログインノードのターミナルからジョブを投入

```
qsub ./myscript.sh
```

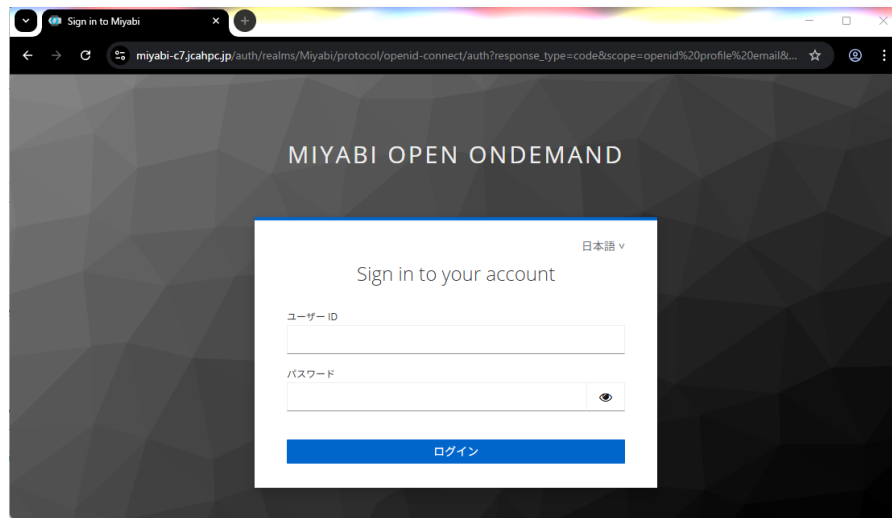
Open OnDemand を使用したMATLAB の利用

Open OnDemand を使用したMATLAB の利用

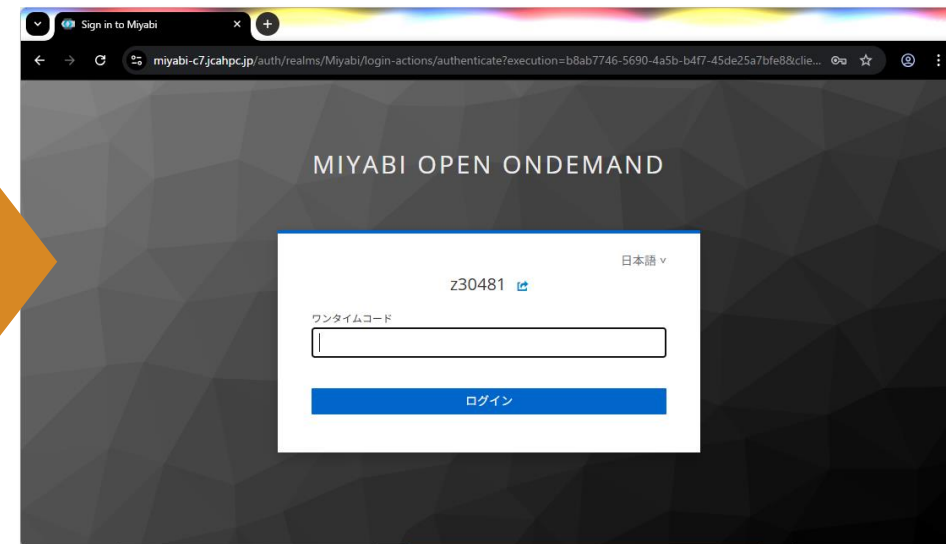
Open OnDemand： オハイオ大学のスーパーコンピューターセンターが開発した、HPC のリソースとインタラクティブなアプリケーションにアクセスできるWeb ベースのポータル

Miyabi-c のOOD のURL にアクセスし、Miyabi
のアカウントでログイン

<https://miyabi-c7.jcahpc.jp/>



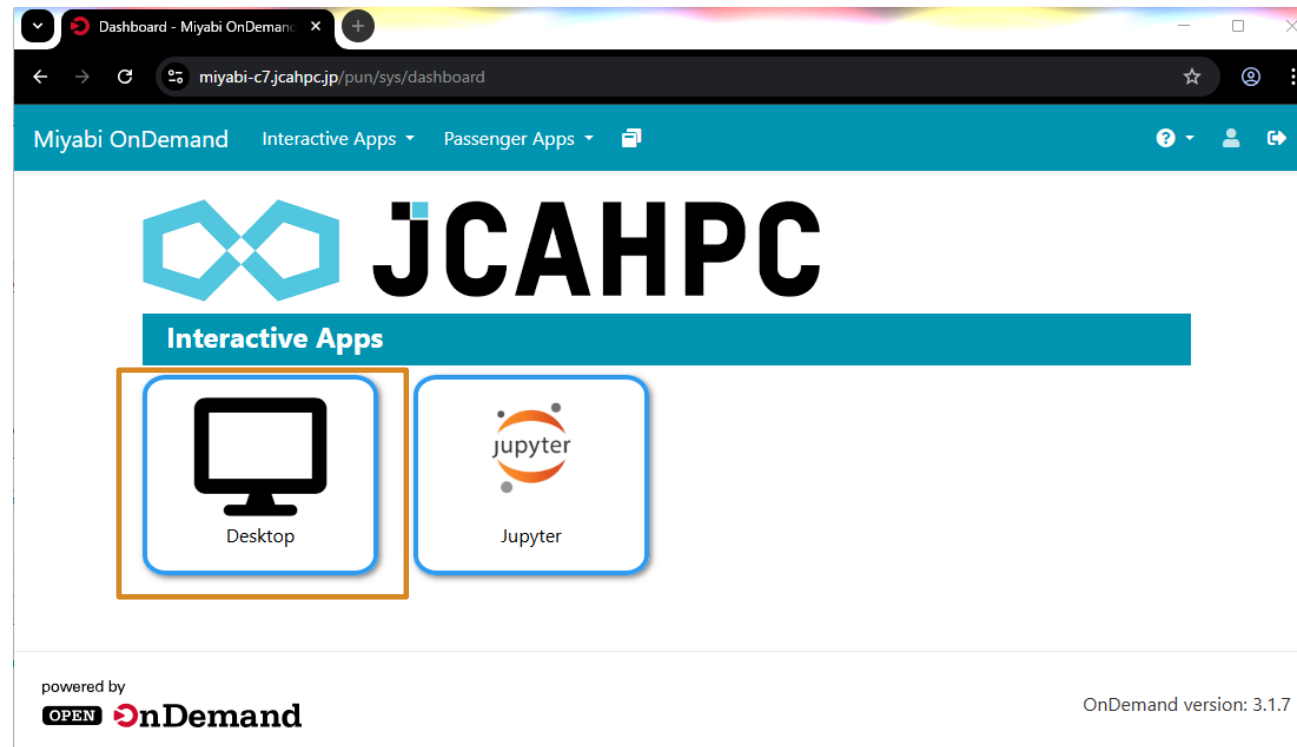
ワンタイムパスワードを入力します



詳細は「Miyabi システム利用手引書」の8章「Open OnDemand」をご覧ください

Open OnDemand を使用したMATLAB の利用

- Interactive Apps から「Desktop」をクリックします



Open OnDemand を使用したMATLAB の利用

- Desktop Environment を任意の項目から選択し、Elapsed time を適切に選択して「Launch」ボタンをクリックします

Miyabi OnDemand Interactive Apps Passenger Apps

Home / My Interactive Sessions / Desktop

Interactive Apps

Desktop

Jupyter

Desktop

This app will launch an interactive desktop on one prepost node.

Desktop Environment

xfce

Elapsed time (1 - 120 hours)

1

☐ I would like to receive an email when the session starts

Launch

* The Desktop session data for this session can be accessed under the [data root directory](#).

powered by **OPEN OnDemand**

OnDemand version: 3.1.7

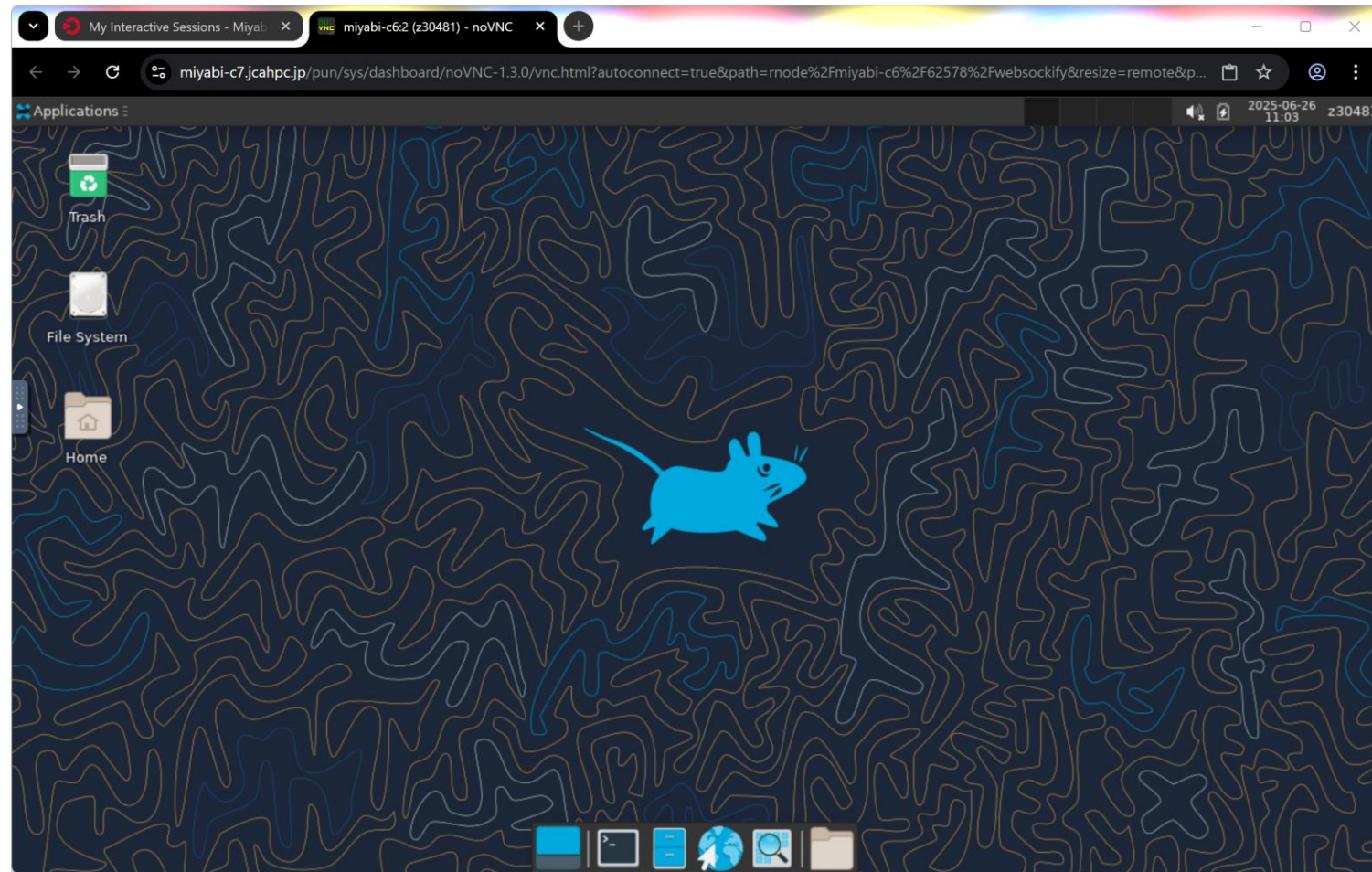
Open OnDemand を使用したMATLAB の利用

- 「Launch Desktop」をクリックします

The screenshot displays the Open OnDemand web interface. At the top, a teal header bar contains the text "Miyabi OnDemand" and navigation links for "Interactive Apps" and "Passenger Apps". A green notification banner at the top center states "Session was successfully created." Below this, a breadcrumb trail shows "Home / My Interactive Sessions". On the left, a sidebar titled "Interactive Apps" lists "Desktop" and "Jupyter". The main content area features a card for a "Desktop (546039.opbs)" session, which is currently "Running" with "1 node" and "7 cores". The card includes details such as the host "miyabi-c6", creation time "2025-06-26 10:58:20 JST", time remaining "1 hour", and session ID "22328846-bae1-4ce2-ad3d-cc7a65a51874". It also has sliders for "Compression" and "Image Quality", both ranging from 0 (low) to 9 (high). A blue "Launch Desktop" button is highlighted with an orange border, and a red "Delete" button is also visible. At the bottom right of the card is a "View Only (Share-able Link)" button. The footer of the interface shows "powered by OPEN OnDemand" and "OnDemand version: 3.1.7".

Open OnDemand を使用したMATLAB の利用

- Web ブラウザーでMiyabi-c のデスクトップ環境が表示されます



Open OnDemand を使用したMATLAB の利用

MATLAB の起動

- デスクトップの下アイコンをクリックしてターミナルを起動します



- /work にディレクトリを作成し、matlab モジュールをロードしてからMATLAB を起動します

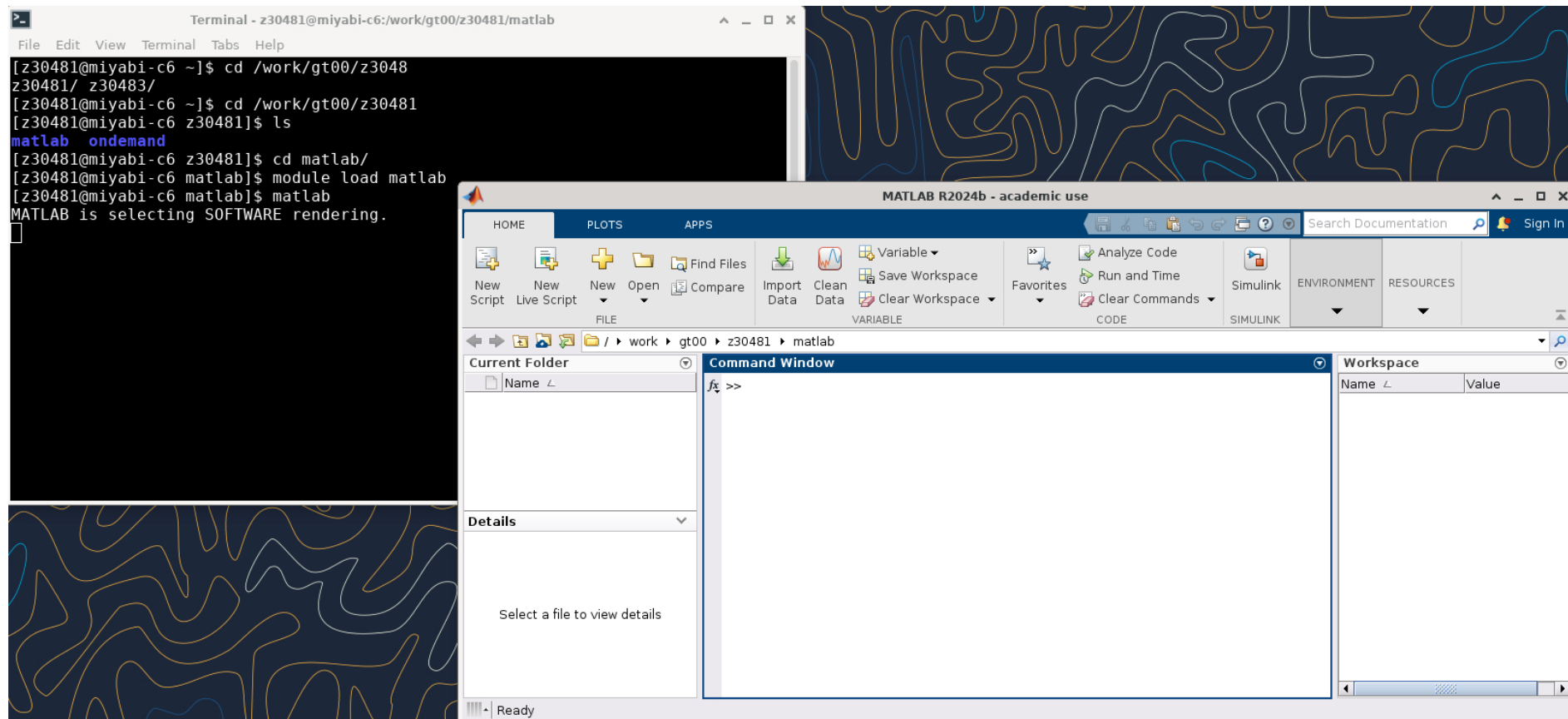
```
cd /work/<groupname>/<username>  
mkdir ./matlab  
cd matlab  
module load matlab  
matlab
```

グレーはターミナルで実行するコマンドです

Open OnDemand を使用したMATLAB の利用

MATLAB の起動

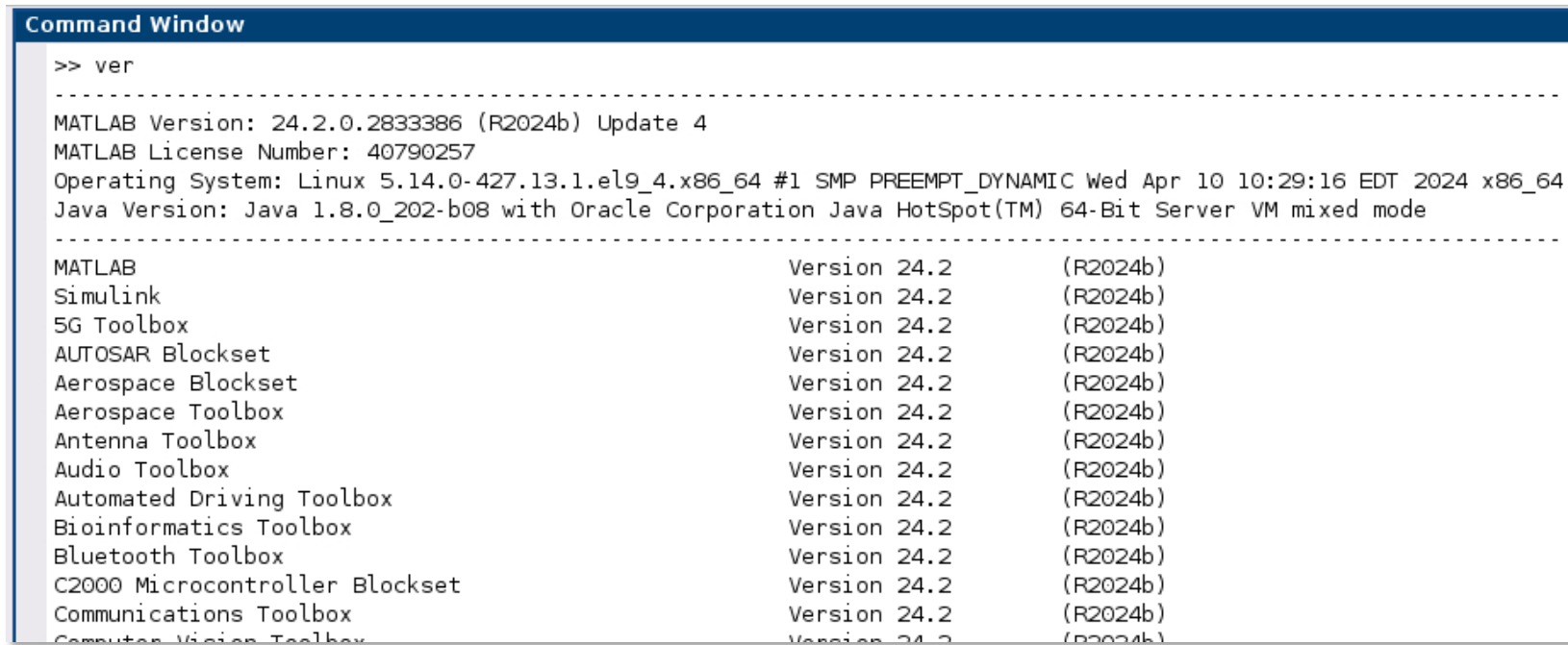
- MATLAB が起動します



Open OnDemand を使用したMATLAB の利用

- MATLAB のコマンドウィンドウで動作を確認します

```
>> ver
```



Command Window

```
>> ver
```

MATLAB Version: 24.2.0.2833386 (R2024b) Update 4
MATLAB License Number: 40790257
Operating System: Linux 5.14.0-427.13.1.el9_4.x86_64 #1 SMP PREEMPT_DYNAMIC Wed Apr 10 10:29:16 EDT 2024 x86_64
Java Version: Java 1.8.0_202-b08 with Oracle Corporation Java HotSpot(TM) 64-Bit Server VM mixed mode

MATLAB	Version 24.2	(R2024b)
Simulink	Version 24.2	(R2024b)
5G Toolbox	Version 24.2	(R2024b)
AUTOSAR Blockset	Version 24.2	(R2024b)
Aerospace Blockset	Version 24.2	(R2024b)
Aerospace Toolbox	Version 24.2	(R2024b)
Antenna Toolbox	Version 24.2	(R2024b)
Audio Toolbox	Version 24.2	(R2024b)
Automated Driving Toolbox	Version 24.2	(R2024b)
Bioinformatics Toolbox	Version 24.2	(R2024b)
Bluetooth Toolbox	Version 24.2	(R2024b)
C2000 Microcontroller Blockset	Version 24.2	(R2024b)
Communications Toolbox	Version 24.2	(R2024b)
Computer Vision Toolbox	Version 24.2	(R2024b)

実行結果

水色はMATLAB のコマンドウィンドウで実行するコマンド

Open OnDemand を使用したMATLAB の利用

- クラスタプロファイルと連携スクリプトの設定をします

```
>> run('/work/opt/local/x86_64/apps/matlab/R2024b/toolbox/local/pluginPBS/configCluster')
```

Command Window

```
>> run('/work/opt/local/x86_64/apps/matlab/R2024b/toolbox/local/pluginPBS/configCluster')  
Warning: Unable to write to requested folder '/work/opt/local/x86_64/apps/matlab/R2024b/toolbox/local/pluginPBS'. Perhaps you do not have the  
correct access permissions. You will be unable to store any job and tasks here.  
> In parallel.internal.cluster/FileStorage (line 60)  
In parallel.internal.cluster.FileStorage.createFileStorage (line 112)  
In parallel.internal.cluster.StorageFactory.createStorage (line 25)  
In parallel.cluster/Generic (line 332)  
In configCluster>assembleClusterProfile (line 261)  
In configCluster (line 217)  
In run (line 112)  
Complete. Default cluster profile set to "Miyabi_R2024b".  
fx >> |
```

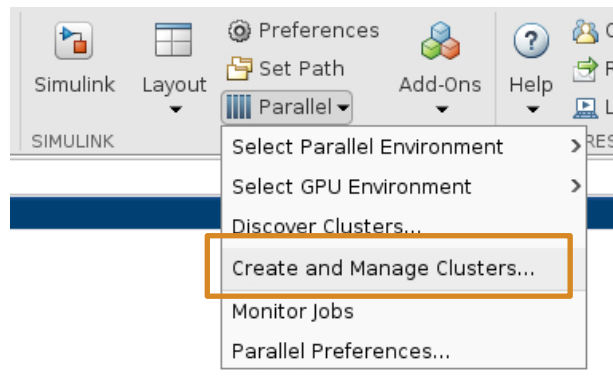
実行結果

警告が出ますが問題はありません

Open OnDemand を使用したMATLAB の利用

クラスタープロファイルの設定

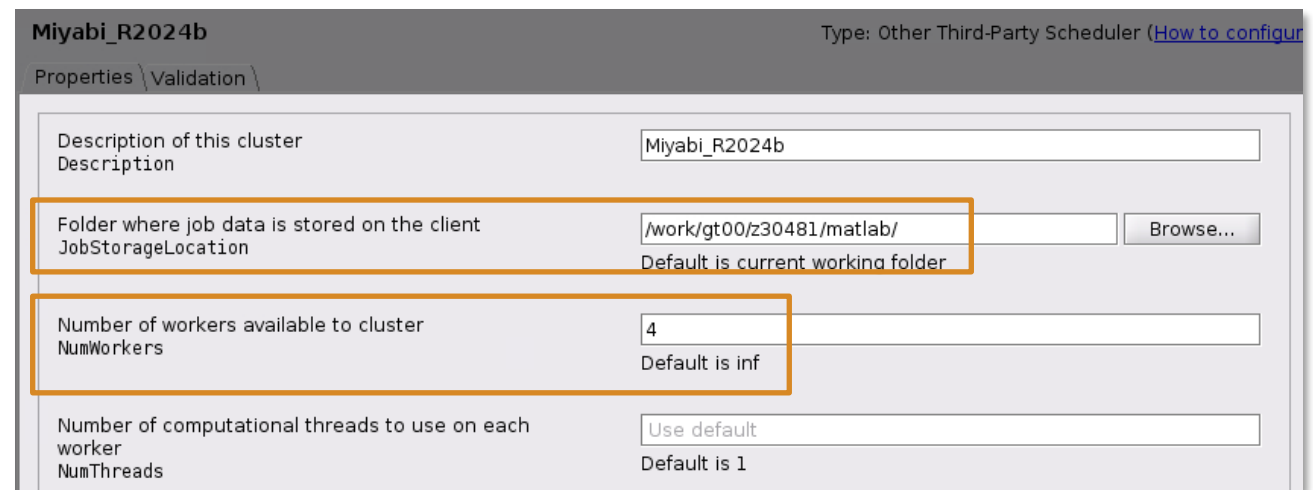
ホームタブのメニューの「Parallel」→
「Create and Manage Clusters...」を
クリックします



Miyab_R2024b のプロファイルのデフォルトから設定を変更
します

JobStorageLocation、NumWorkers の変更

`/work/<groupname>/<username>/matlab/`



Open OnDemand を使用したMATLAB の利用

クラスタープロファイルの設定

Miyab_R2024b のプロファイルのデフォルトから設定を変更します

GroupName、QueueName、WallTime の変更

/work/<groupname>/<username>/matlab/

Name	Value	Type
EnableDebug	false	Logical
GroupName	gt00	String
Nodes	1	Number
QueueName	lecture-c	String
UseSmpd	false	Logical
WallTime	00:10:00	String

Additional properties for plugin scripts
AdditionalProperties

FILES AND FOLDERS

Automatically send code files to cluster. Data files or folders must be listed in the AttachedFiles property.
AutoAttachFiles

Use default
Default is true

Manually specify files and folders to copy from client to cluster nodes (One entry per line)
AttachedFiles

Manually specify folders to add to the workers' search path (One entry per line)
AdditionalPaths

Done Cancel

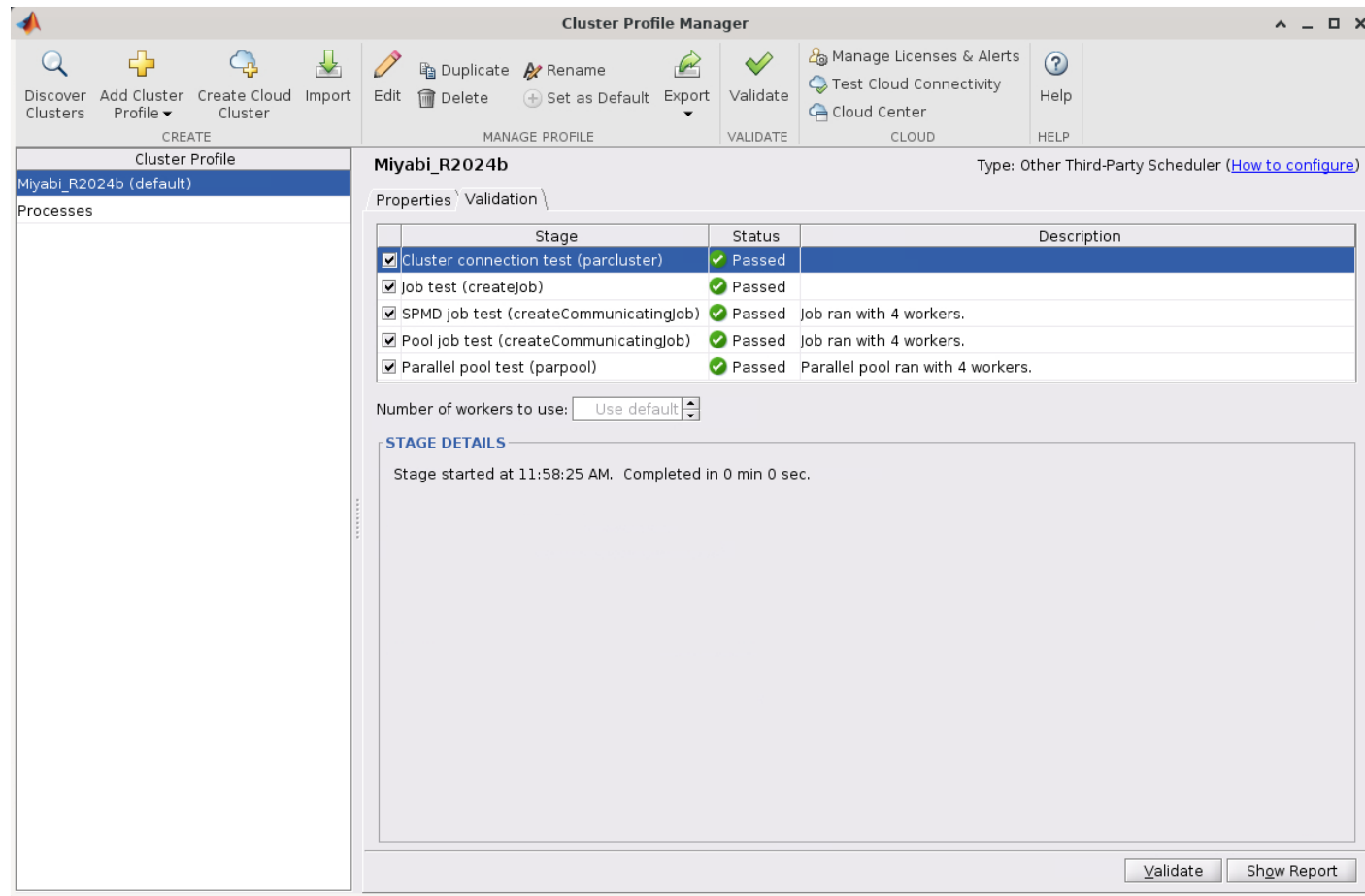
使用できるキューは **lecture-c** または **tutorial-c** (本講習会中のみ) です

編集後は「Done」をクリックします

Open OnDemand を使用したMATLAB の利用

クラスタープロファイルの検証

- 「Validation」タブをクリックして「Validate」ボタンをクリックします



インタラクティブな並列処理

【実習】 MATLAB のMiyabi へのジョブの投入の方法

インタラクティブな並列処理

- MATLAB の並列処理を実行します

```
>> parpool(4)
>> spmd
>> spmdIndex
>> end
```

- 並列処理が終わったらプールを閉じます

```
>> delete(gcp)
```



```
Command Window
>> parpool(4)
Starting parallel pool (parpool) using the 'Miyabi_R2024b' profile ...
Connected to parallel pool with 4 workers.

ans =

ClusterPool with properties:

    Connected: true
    NumWorkers: 4
        Busy: false
    Cluster: Miyabi_R2024b (Generic Cluster)
    AttachedFiles: {}
    AutoAddClientPath: true
    FileStore: [1x1 parallel.FileStore]
    ValueStore: [1x1 parallel.ValueStore]
    IdleTimeout: 30 minutes (30 minutes remaining)
    SpmdEnabled: true
    EnvironmentVariables: {}

>> spmd
spmdIndex
end
Worker 1:

ans =

    1

Worker 2:

ans =

    2

Worker 3:

ans =

    3

Worker 4:

ans =

    4
```

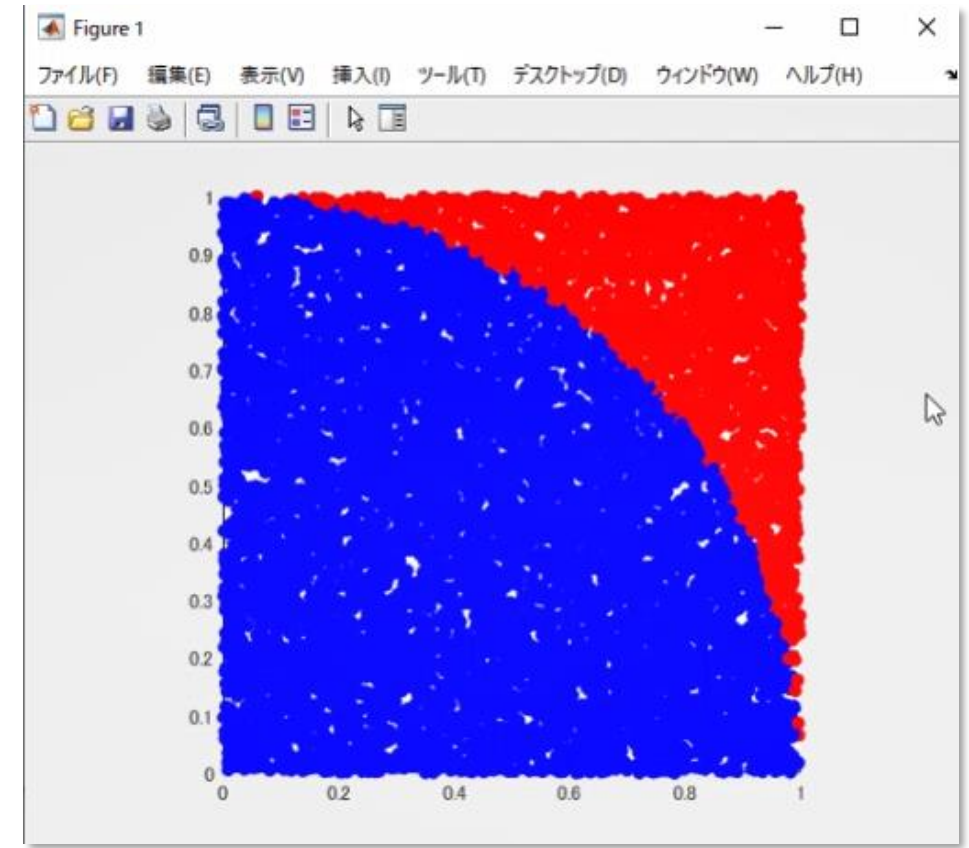
実行結果

【実習】 MATLAB のMiyabi へのジョブの投入の方法

モンテカルロ法 (モンテカルロ・シミュレーション)

並列for 文で計算

```
parfor n = 1:iter 乱数
    x = R * rand(1);
    y = R * rand(1);
    if x^2 + y^2 < R^2
        count(n, 1) = 1;
    end
end
```



円周率 $\hat{=}$ 円内の点の数 / 全体の点の数

【実習】 MATLAB のMiyabi へのジョブの投入の方法

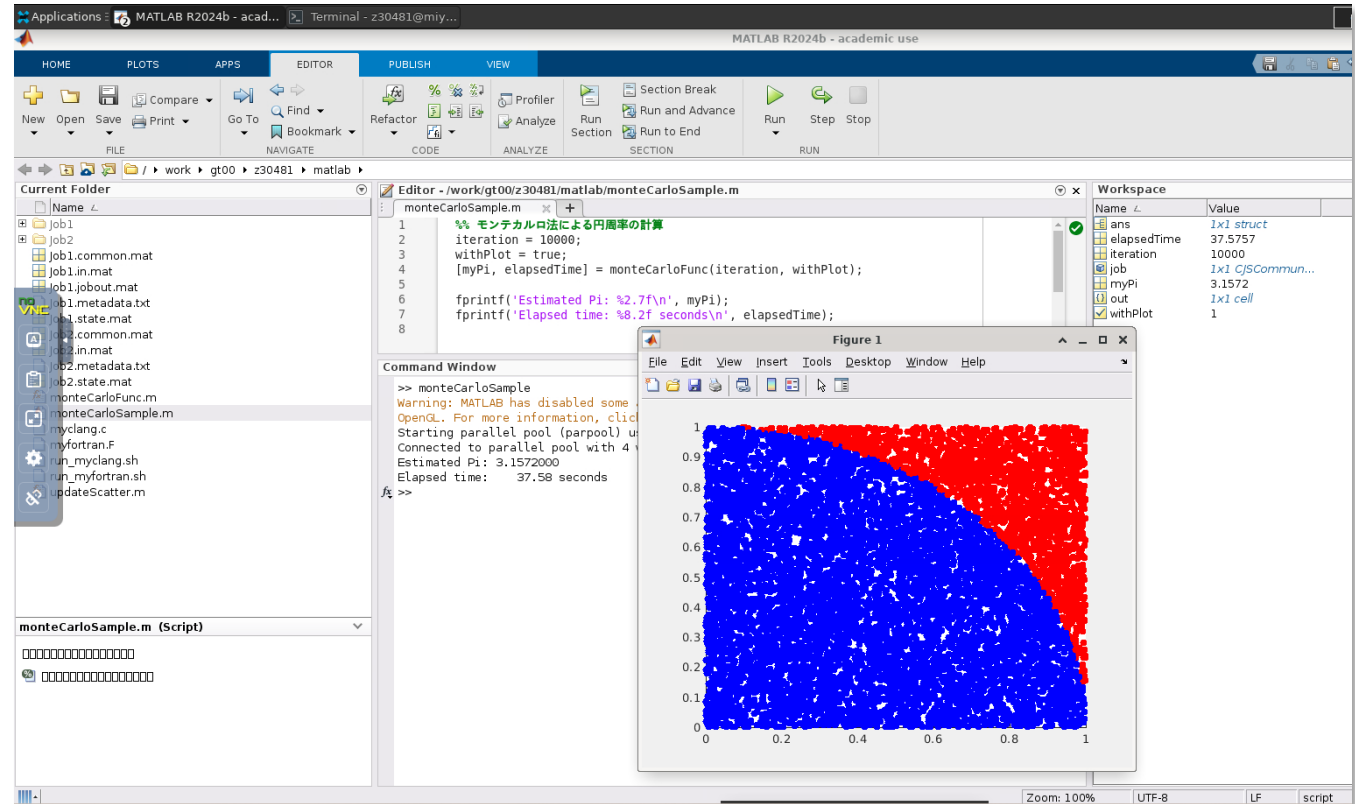
インタラクティブな並列処理

- MATLAB の並列処理を実行します

```
>> monteCarloSample
```

- 並列処理が終わったらプールを閉じます

```
>> delete(gcp)
```



オフロードでの並列処理

【実習】 MATLAB のMiyabi へのジョブの投入の方法

オフロードでの並列処理

- モンテカルロ法の処理をバッチで投入

```
>> job = batch('monteCarloSample', 'Pool', 3)
```

```
Command Window
>> job = batch('monteCarloSample', 'Pool', 3)

job =

Job

Properties:

    ID: 1
    Type: pool
    Username: z30481
    State: running
    StorageBytes: 18883 (19 KB)
    SubmitDateTime: 16-Jul-2025 12:59:17
    StartDateTime:
    RunningDuration: 0 days 0h 0m 0s
    NumWorkersRange: [4 4]
    NumThreads: 1
    SpmdEnabled: true

    AutoAttachFiles: true
    Auto Attached Files: List files
    AttachedFiles: {}
    AutoAddClientPath: true
    AdditionalPaths: /home/z30481/Documents/MATLAB
    FileStore: [1x1 parallel.FileStore]
    ValueStore: [1x1 parallel.ValueStore]
    EnvironmentVariables: {}

Associated Tasks:

    Number Pending: 4
    Number Running: 0
    Number Finished: 0
    Task ID of Errors: []
    Task ID of Warnings: []
    Task Scheduler IDs: 607850.opbs
```

【実習】 MATLAB のMiyabi へのジョブの投入の方法

オフロードでの並列処理

- ステータスを確認

```
>> job.State
```

- ステータスがfinished になっていれば結果を取得

```
>> out = fetchOutputs(job)
```

- 中身を確認

```
>> out{:}
```

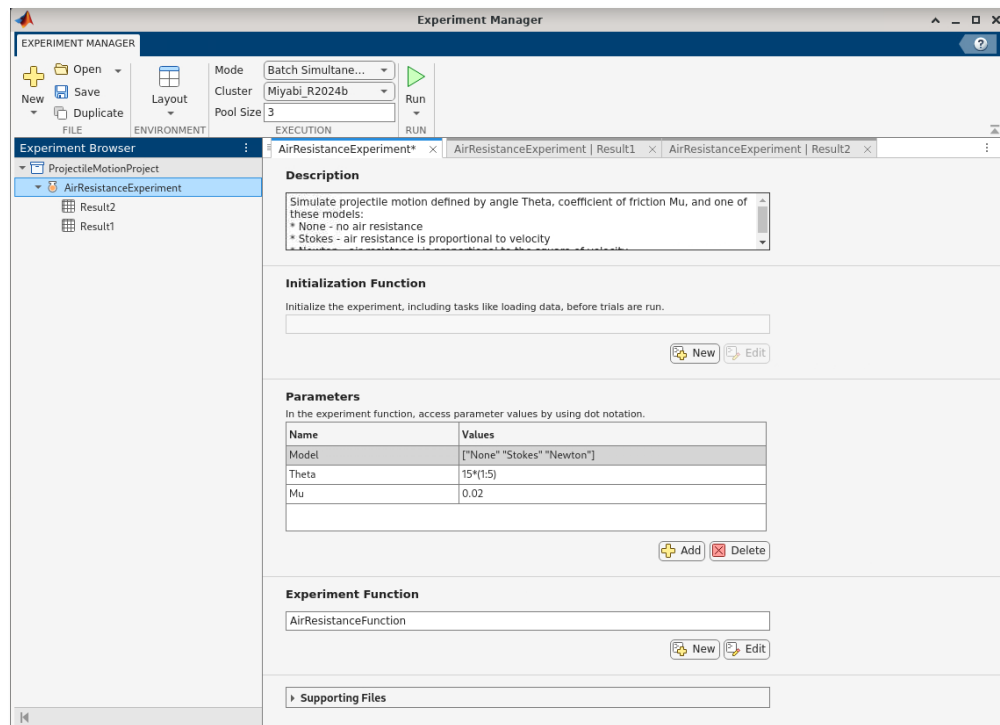
```
>> job.State  
ans =  
    'finished'  
  
>> out = fetchOutputs(job)  
out =  
    1×1 cell array  
        {1×1 struct}  
  
>> out{:}  
ans =  
    struct with fields:  
    elapsedTime: 0.6154  
    iteration: 10000  
    myPi: 3.1652  
    withPlot: 0
```

実験マネージャアプリを使った並列処理

実験マネージャーアプリとは

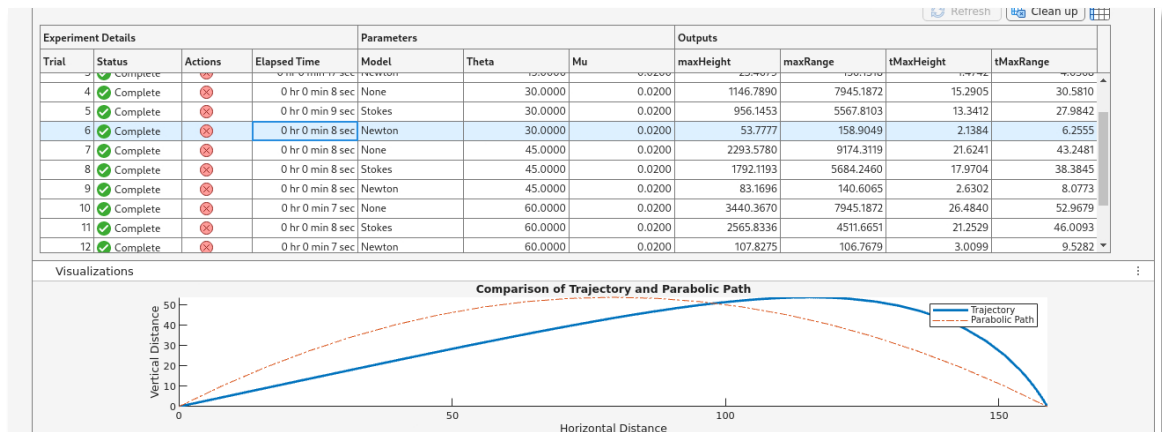
- パラメーター最適化などの実験の管理をおこなうアプリ
インタラクティブまたはバッチでの並列処理も対応

パラメータスウィープやベイズ最適化などに



Experiment Details				Parameters
Trial	Status	Actions	Elapsed Time	Model
1	Running		0 hr 0 min 10 sec	None
2	Running		0 hr 0 min 10 sec	Stokes
3	Running		0 hr 0 min 10 sec	Newton
4	Queued			None
5	Queued			Stokes
6	Queued			Newton
7	Queued			None
8	Queued			Stokes

実行結果やプロットを確認



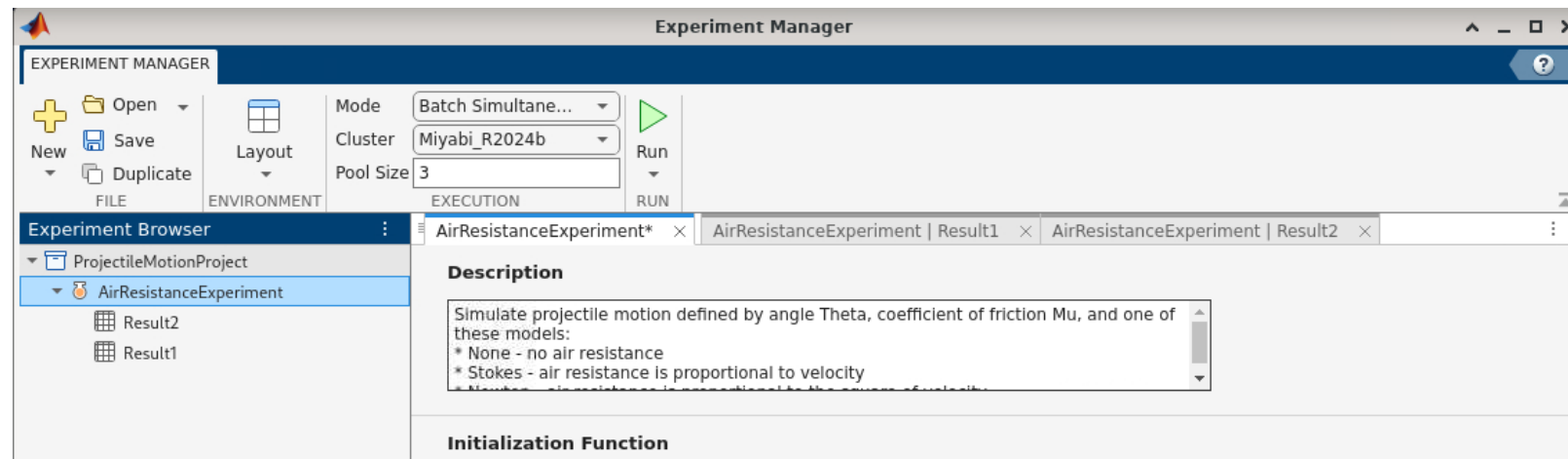
【実習】 MATLAB のMiyabi へのジョブの投入の方法

実験マネージャーからのジョブ投入

- ドキュメント例を表示

```
>> openExample('matlab/ProjectileMotionExperimentExample')
```

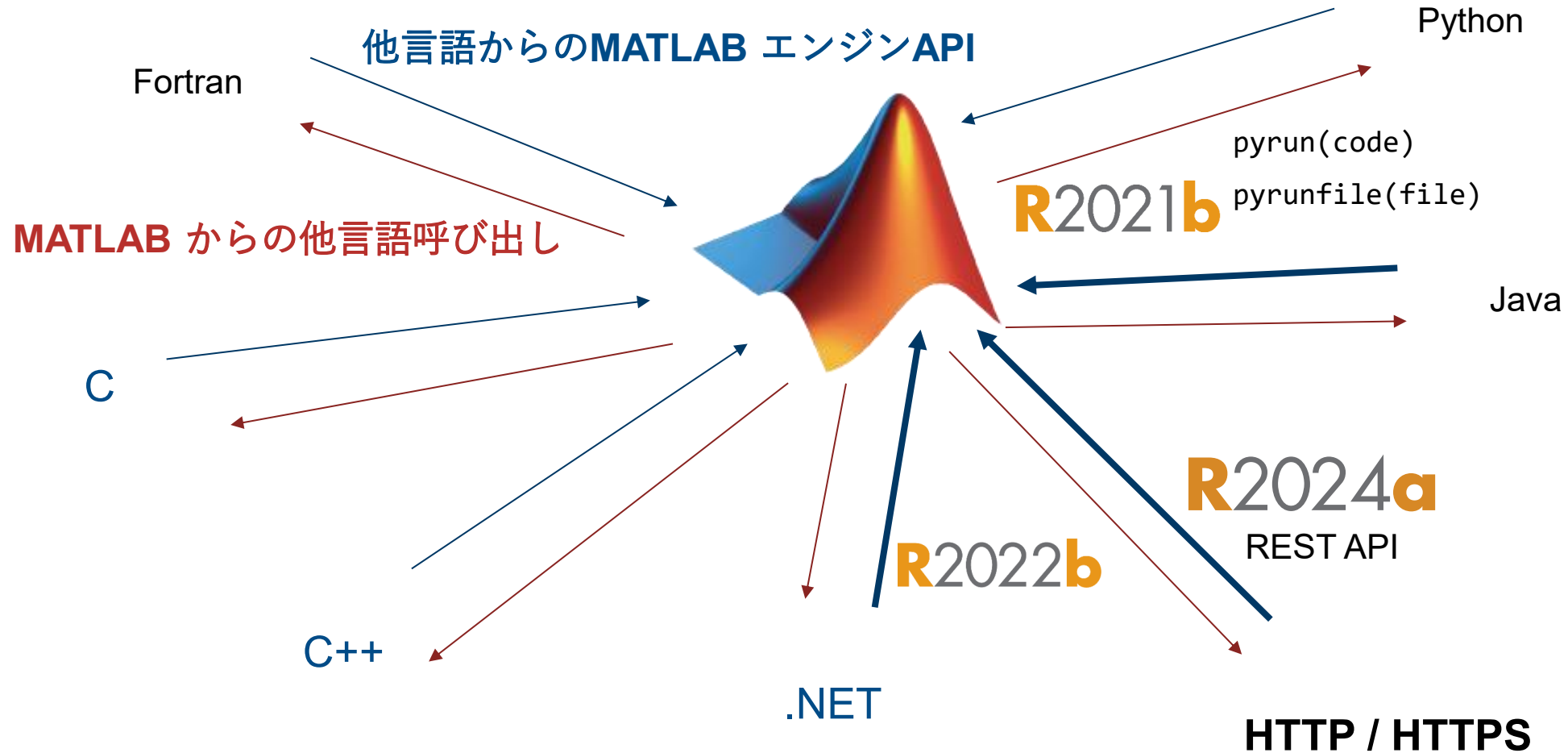
- 実験マネージャーアプリで「Mode」を「Simultaneous」または「Batch_Simultaneous」にして「Cluster」を「Miyabi_R2024b」に指定し、「Run」をクリック



Fortran、CのプログラムからMATLABを呼び出す実演



外部言語インターフェイス



【実習】 Fortran からMATLAB を呼び出す処理を書く

myfortran.F

```
#include "fintrf.h"
C
#if 0
C
C    fengdemo.F
C    .F file need to be preprocessed to generate .for equivalent
C
C    MATLAB エンジン起動
#endif
```

```
ep = engOpen('matlab -nodisplay')
```

```
if (ep .eq. 0) then
    write(6,*) 'Can''t start MATLAB engine'
    stop
endif
```

```
if (engEvalString(ep, 'D = .5.*(-9.8).*T.^2;') .ne. 0) then
    write(6,*) 'engEvalString failed'
    stop
endif
```

自由落下の計算をMATLAB
エンジンで実行

$$D = -\frac{1}{2}gt^2$$

【実習】 Fortran からMATLAB を呼び出す処理を書く

- MATLAB Engine API for Fortran を使ったコードを書きます。本実習では事前に配布した myfortran.F を使用します
- MATLAB を起動し、Fortran コードをコンパイルします
ログインノードのターミナル

```
$ module load gcc
```

MATLAB のコマンドウィンドウ

```
>> mex -client engine myfortran.F
```

Command Window

```
>> mex -client engine myfortran.F  
Building with 'gfortran'.  
MEX completed successfully.  
>> ls myfortran*  
myfortran  myfortran.F
```

実行結果

【実習】 Fortran からMATLAB を呼び出す処理を書く

- ジョブ投入のためのスクリプトを書きます

シェルスクリプト (**run_myfortran.sh**) を作成

```
#!/bin/bash
#PBS -q lecture-c ← 使用できるキューは lecture-c または tutorial-c (本講習会中のみ)です
#PBS -l select=1
#PBS -W group_list=gt00
#PBS -j oe
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/work/opt/local/x86_64/
apps/matlab/R2024b/sys/os/glnxa64

cd ${PBS_O_WORKDIR}
module load gcc matlab
./myfortran
```

GLIBCXX_3.4.30 not foundの
エラーを回避するため

- ジョブを投入します

```
$ qsub ./run_myfortran.sh
```

実行結果

```
[z30481@miyabi-c6 matlab]$ qsub ./run_myfortran.sh
607969.opbs
```

ジョブID

【実習】 Fortran からMATLAB を呼び出す処理を書く

- permission denied のエラーがでた場合は実行権限を確認ください

実行権限(x)が無い場合は権限を付与

```
-rw-r----- 1 z30481 gt00 128 Nov 6 12:18 run_myclang.sh  
-rw-r----- 1 z30481 gt00 129 Nov 6 15:43 run_myfortran.sh
```

chmod u+x *.sh

```
-rwxr----- 1 z30481 gt00 128 Nov 6 12:18 run_myclang.sh  
-rwxr----- 1 z30481 gt00 129 Nov 6 15:43 run_myfortran.sh
```

chmod u+x myfortran

```
-rwxr----- 1 z30481 gt00 22744 Nov 6 12:18 myfortran
```

【実習】 Fortran からMATLAB を呼び出す処理を書く

- ジョブのステータスを確認します

```
$ qstat
```

実行結果

```
[z30481@miyabi-c6 matlab]$ qstat
Miyabi scheduled stop time: 0000/00/00(week) 00:00:00 (Remain: 00:00:00)

JOB_ID      JOB_NAME  STATUS   PROJECT  QUEUE      START_DATE  ELAPSE      TOKEN  NODE  MIG
607838      desktop  RUNNING  gt00     prepost-d  07/16 12:53:35 01:16:13    -     1    -
607969      run_myfort  RUNNING  gt00     lecture-c  07/16 14:10:10 00:00:10    -     1    -
```

ジョブ実行中

```
[z30481@miyabi-c6 matlab]$ qstat 607969
Miyabi scheduled stop time: 0000/00/00(week) 00:00:00 (Remain: 00:00:00)
No unfinished job found.
```

実行結果

- 終了したらジョブの結果(.out)とエラー(.err)を確認します

```
$ cat *<jobid>.out
$ cat *<jobid>.err
```

```
[z30481@miyabi-c6 matlab]$ cat *607969
MATLAB computed the following distances:
time(s)  distance(m)
1.00     -4.90
2.00     -19.6
3.00     -44.1
4.00     -78.4
5.00     -123.
6.00     -176.
7.00     -240.
8.00     -314.
9.00     -397.
10.0     -490.
```

【実習】 C言語からMATLAB を呼び出す処理を書く

myclang.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "engine.h"
#define BUFSIZE 256

int main()
{
    Engine *ep;
    mxArray *T = NULL, *result = NULL;
    char buffer[BUFSIZE+1];
    double time[11] = { 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0 };

    /*
     * Call engOpen with a NULL string. This starts a MATLAB process
     * on the current host using the command "matlab".
     */
    if (!(ep = engOpen(""))) {
        fprintf(stderr, "\nCan't start MATLAB engine\n");
        return EXIT_FAILURE;
    }
}
```

MATLAB エンジン起動

```
/*
 * Create a variable for the data
 */
T = mxCreateDoubleMatrix(1, 11, mxREAL);
memcpy((void *)mxGetPr(T), (void *)time, sizeof(time));
/*
 * Place the variable T into the MATLAB workspace
 */
engPutVariable(ep, "T", T);

/*
 * Evaluate a function of time, distance = (1/2)g.*t.^2
 * (g is the acceleration due to gravity)
 */
engEvalString(ep, "D = .5.*(-9.8).*T.^2;");

result = engGetVariable(ep, "D");
double* resultD = mxGetPr(result);
for (int i=0; i<11; i++){
    printf("%1f\t%1f\n", time[i], resultD[i]);
}
engClose(ep);

return EXIT_SUCCESS;
}
```

自由落下の計算をMATLAB
エンジンで実行

$$D = -\frac{1}{2}gt^2$$

【実習】 C言語からMATLAB を呼び出す処理を書く

- MATLAB Engine API for C を使ったコードを書きます。本実習では事前に配布した `myclang.c` を使用します
- MATLABを起動し、 C コードをコンパイルします

MATLAB のコマンドウィンドウ

```
>> mex -client engine myclang.c
```

実行結果

Command Window

```
>> mex -client engine myclang.c  
Building with 'gcc'.  
MEX completed successfully.
```

【実習】 C言語からMATLAB を呼び出す処理を書く

- ジョブ投入のためのスクリプトを書きます

シェルスクリプト (**run_myclang.sh**) を作成

```
#!/bin/bash
#PBS -q lecture-c
#PBS -l select=1
#PBS -W group_list=gt00
#PBS -j oe
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/work/opt/local/x86_64/
apps/matlab/R2024b/sys/os/glnxa64

cd ${PBS_O_WORKDIR}
module load matlab
./myclang
```

→ 使用できるキューは **lecture-c** または **tutorial-c** (本講習会中のみ)です

GLIBCXX_3.4.30 not foundの
エラーを回避するため

- ジョブを投入します

```
$ qsub ./run_myclang.sh
```

実行結果

```
[z30481@miyabi-c6 matlab]$ qsub ./run_myclang.sh
608000.opbs
```

ジョブID

【実習】 C言語からMATLAB を呼び出す処理を書く

- ジョブのステータスを確認します

```
$ qstat
```

実行結果

```
[z30481@miyabi-c6 matlab]$ qstat
Miyabi scheduled stop time: 0000/00/00(week) 00:00:00 (Remain: 00:00:00)

JOB_ID      JOB_NAME    STATUS    PROJECT  QUEUE      START_DATE  ELAPSE      TOKEN  NODE  MIG
607838      desktop    RUNNING   gt00     prepost-d   07/16 12:53:35 01:36:21    -      1    -
608000      run_myclan  RUNNING   gt00     lecture-c   07/16 14:29:47 00:00:10    -      1    -
```

ジョブ実行中

```
[z30481@miyabi-c6 matlab]$ qstat 608000
Miyabi scheduled stop time: 0000/00/00(week) 00:00:00 (Remain: 00:00:00)
No unfinished job found.
```

実行結果

ジョブ終了

- 終了したらジョブの結果(.out) とエラー(.err) を確認します

```
$ cat *<jobid>.out
$ cat *<jobid>.err
```

```
[z30481@miyabi-c6 matlab]$ cat *608000
0.000000      -0.000000
1.000000      -4.900000
2.000000     -19.600000
3.000000     -44.100000
4.000000     -78.400000
5.000000    -122.500000
6.000000    -176.400000
7.000000    -240.100000
8.000000    -313.600000
9.000000    -396.900000
10.000000   -490.000000
```

Open OnDemand を使用したMATLAB の利用

- 利用終了後は「Delete」ボタンをクリックします

The screenshot displays the 'My Interactive Sessions' page in the Open OnDemand interface. At the top, a breadcrumb trail shows 'Home / My Interactive Sessions'. On the left, a sidebar titled 'Interactive Apps' contains two options: 'Desktop' (selected) and 'Jupyter'. The main content area shows a single session card for 'Desktop (546039.opbs)'. The card indicates the session is 'Completed' with a refresh icon. It lists the creation time as '2025-06-26 10:58:20 JST' and the 'Session ID' as '22328846-bae1-4ce2-ad3d-cc7a65a51874'. A red 'Delete' button with a white 'x' icon is positioned to the right of the session details. At the bottom of the card, a note states: 'For debugging purposes, this card will be retained for 6 more days'.

関連リソース

- [Miyabi の利用支援ポータル](#)から
「ドキュメント閲覧」→「**Miyabi MATLAB**ジョブ連携手順書」
- MathWorks 公式ドキュメント
 - <https://jp.mathworks.com/help/>
- 東大MATLAB アンバサダー (主に東大の学生向けサポート)
 - https://sites.google.com/view/ut-matlab-amb/MATLAB_QA



© 2025 The MathWorks, Inc. MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.