

第250回お試しアカウント付き並列プログラミング講習会 「MATLABの実行方法」

2025年7月30日（水） 13：00 - 16：00

第一部 MATLAB入門とその高速化入門

13:00 - 14:00 MATLAB概要と高速化入門

- MATLAB概要
- 高速化のためのコードの書き方
 - PC単体でもできるTips
 - スーパーコンピュータとの連携

第二部 スーパーコンピュータでMATLABを動かすハンズオン実習

14:00 - 16:00 MATLABを起動してサンプルデータを用いた演習

- MATLABの並列処理概要
- 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
- Fortran, CのプログラムからMATLABを呼び出す実演

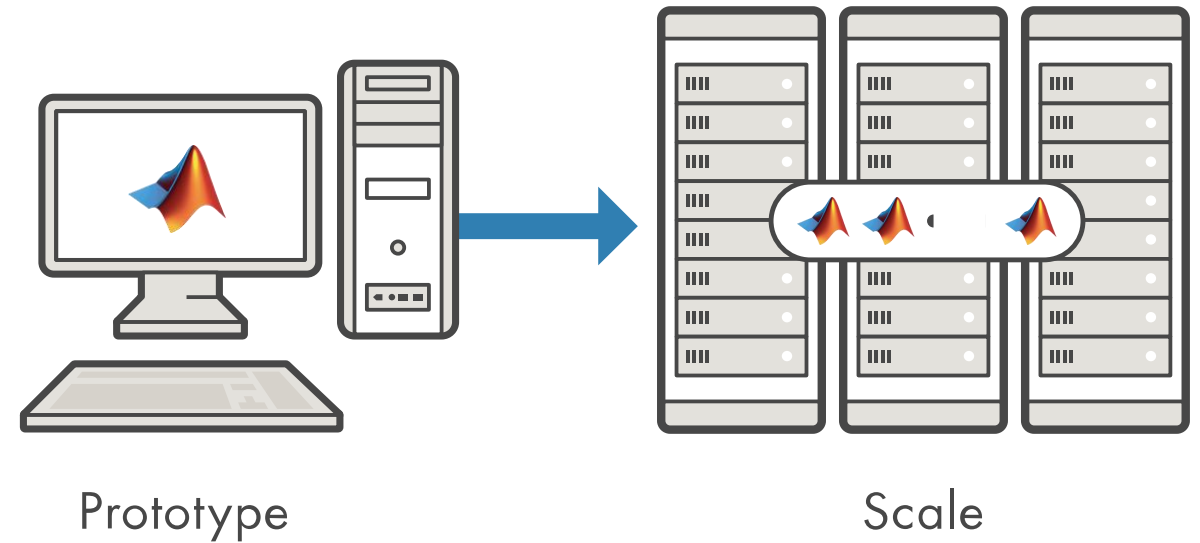
MATLAB相談会(現地参加者のみ) 16:30-17:00

MATLAB概要と高速化入門

2025年7月30日（水）13：00 - 14：00

MathWorks Japan

安川 朋昭（カスタマーサクセスエンジニア）



プログラム

1. MATLAB概要
2. 高速化のためのコードの書き方
 - PC単体でもできるTips
 - スーパーコンピュータとの連携
3. Q & A



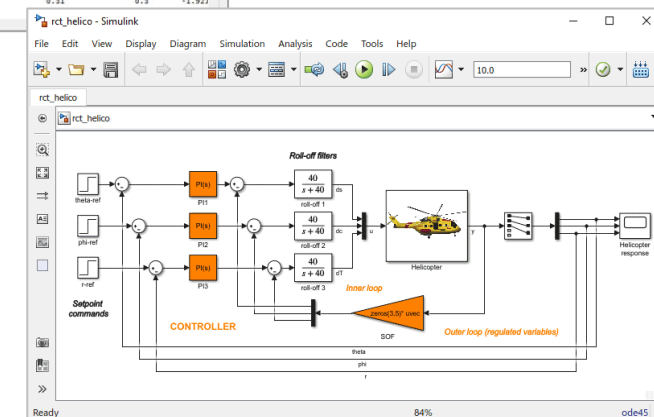
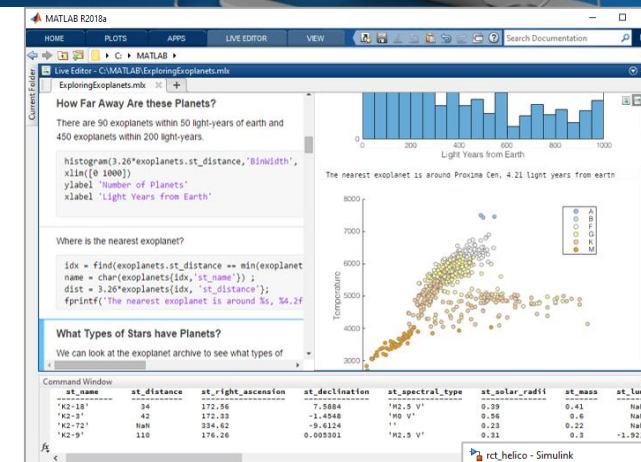
MATLAB概要

MATLAB® & SIMULINK®



Accelerating the pace of Engineering & Science

- MATLAB：アルゴリズム開発、データ解析、可視化、数値計算のためのプログラミング統合開発環境
- Simulink：システムの設計やシミュレーション、実装、テストのためのグラフィカル実行環境
- 用途に特化したアドオン100 製品以上を提供



業界標準のToolboxで教育・研究を効率化

Automated Driving Toolbox

ADAS および自動運転システムの設計、シミュレーション、およびテスト



[アドオン製品一覧](#)

5G Toolbox

5G 通信システムや 5G-Advanced 通信システムのシミュレーション、解析、テスト



Computer Vision Toolbox

コンピュータービジョン、3D ビジョン、および動画処理システムの設計およびテスト



Medical Imaging Toolbox

2D および 3D 医用画像の可視化、レジストレーション、セグメント化、およびラベル付け



Deep Learning Toolbox

ディープラーニング ネットワークの設計、学習、解析



Text Analytics Toolbox

テキストデータの解析とモデル化



MATLAB および Simulink を使用している業界



航空宇宙および防衛



自動車



生物科学



生命工学および製薬



通信



エレクトロニクス



エネルギー生産



金融サービス



産業機械



医療機器



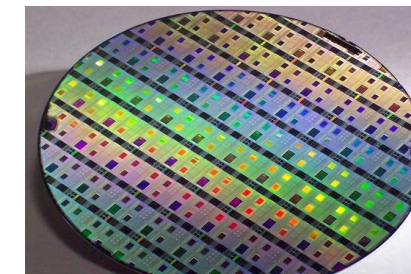
金属、材料、鉱業



神経科学



鉄道システム



半導体

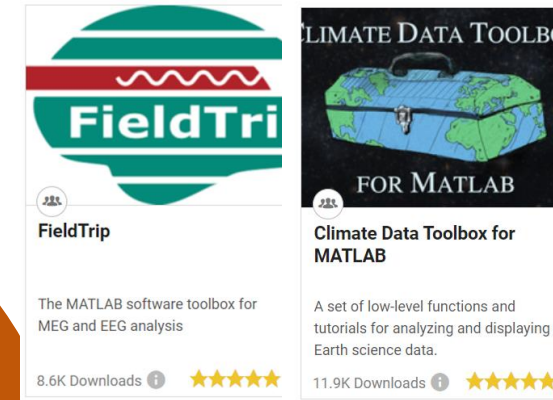


ネット

科学の再現性を高めるための取り組み（オープンサイエンス）

ドキュメント

オープンソース
コミュニティツール



GitHub
Open in MATLAB Online

- 200,000+ コードリポジトリ
- オンラインでのMATLAB実行

CODE OCEAN



オープンサイエンス
の支援

再現性のあるサイエンスのた
めのコードとデータの共有

科学研究の支援

- ファンディング
- 業界パートナー
- 技術的な専門知識

MathWorks、国立情報学研究所とオープンサイエンス推進で協力 クラウドベースの研究データ管理プラットフォーム「GakuNin RDM」からブ ラウザ経由でMATLAB にアクセス可能に

Share

X ポスト

Share

リンク貼っておく

Tokyo, Japan - (2024 年 3 月 4 日)

MathWorksは本日、国立情報学研究所NII（読み：エヌアイアイ）、所長：黒橋 禎夫、東京都千代田区）との協力の下、NIIの研究データ管理プラットフォーム「GakuNin RDM」からブラウザ経由でMATLABにアクセスできる機能を提供することを発表しました。

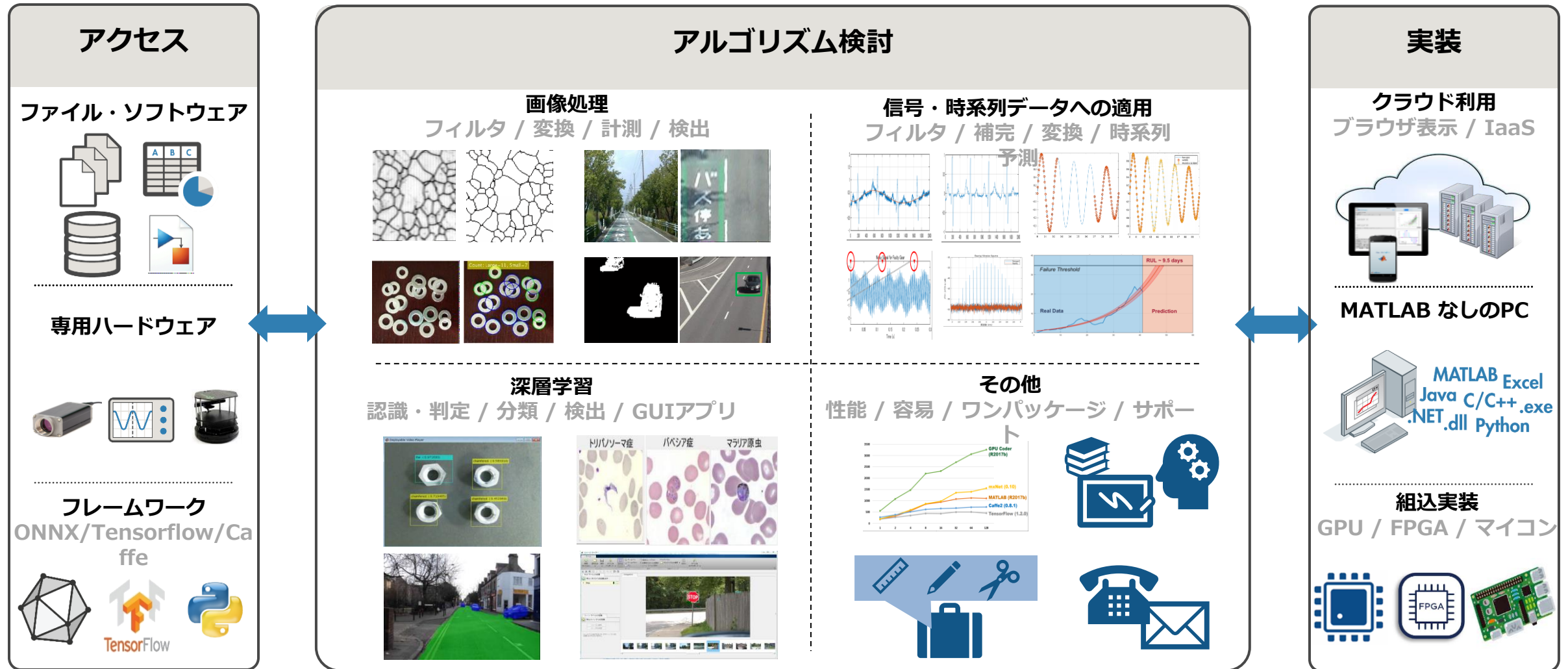
これにより、研究者はNIIが提供するクラウド上でMATLABコードを実行し、解析結果をGakuNin RDMプラットフォーム上に保存して共有できるようになります。また、プラットフォーム間の連携設定や切り替えなどを行うことなく、データの保存、共有、解析がシームレスに行えるようになります。再現性の担保も容易になり、研究データへのアクセスを適切に管理しつつ解析を行うことが可能です。

利用者のメリット

研究者は、GakuNin RDMから直接クラウド上のMATLABにアクセスすることができるようになるため、自身のコンピュータにソフトウェアをインストールすることなく、研究データの解析と共有を行うことができ、研究者同士の連携が円滑になります。また、研究データの管理と解析を一元化することにより、研究プロセスの効率化だけでなく、研究の再現性も向上します。

なお、このMATLAB環境の利用にはMathWorks製品のライセンスが必要です。日本国内の数多くの高等教育機関が導入している全学利用向けのCampus-Wide Licenseを利用することで、さらに手軽にアクセスできます。

MATLABの特徴：データアクセスから実装まで



他言語との連携

Interface:

MATLAB から他言語を呼ぶ



Engine:

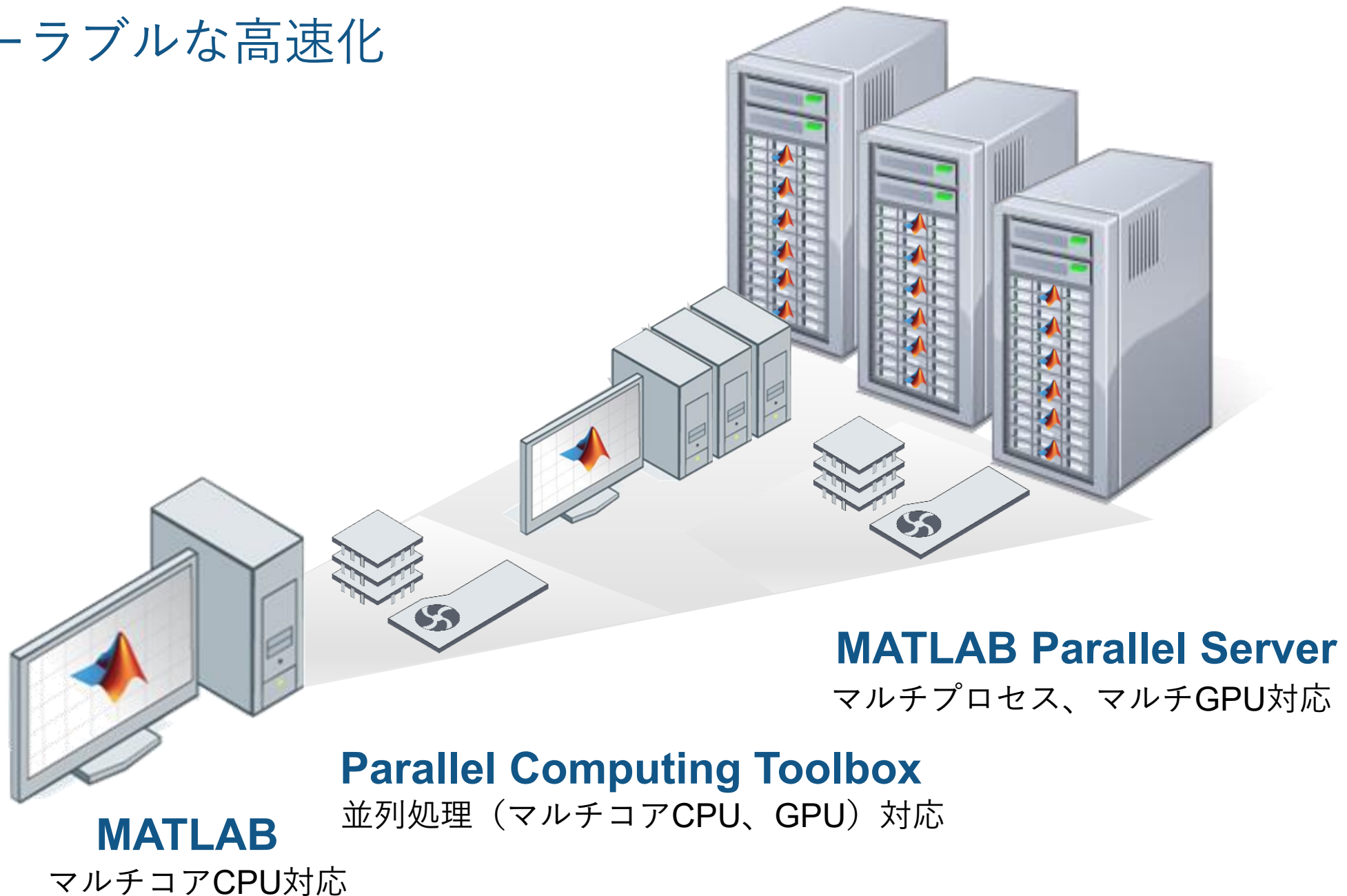
他言語から MATLAB を呼ぶ



- Java
- Python*
- C
- C++
- Fortran
- COM コンポーネント & ActiveX® controls
- RESTful, HTTP, & WSDL web

- Java
- Python*
- C/C++
- .NET
- Fortran
- COM Automation server

スケーラブルな高速化



高速化のためのコードの書き方

～処理が遅い理由を見つける～

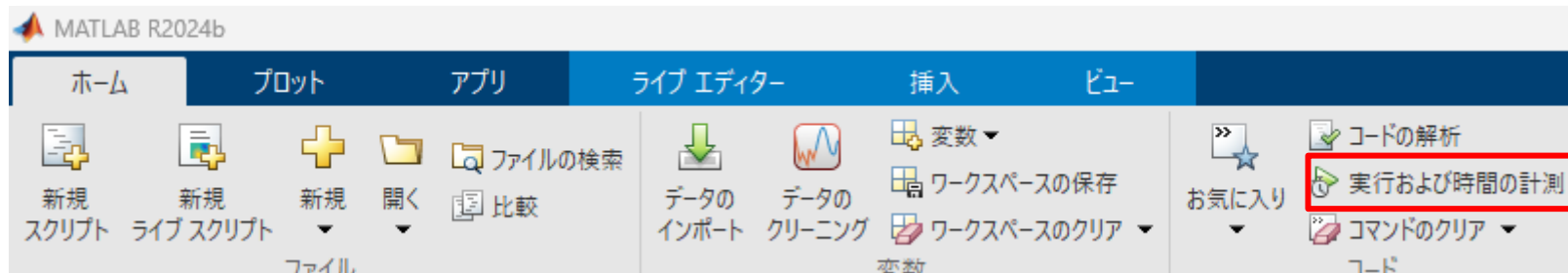


処理が遅い理由は何？

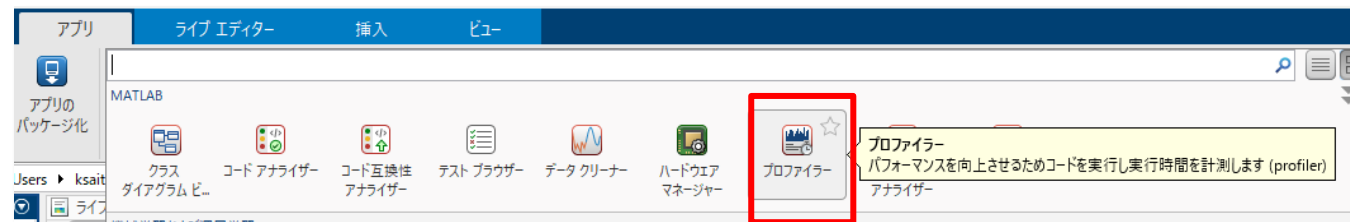
```
airlineData = readtable("airlinesmall.csv");
countArray = zeros(22, 12);
for n = 1:height(airlineData)
    for y = 1987:2008
        yy = y-1986;
        for m = 1:12
            if airlineData.Year(n) == y
                if airlineData.Month(n) == m
                    countArray(yy, m) = countArray(yy, m) + 1;
                end
            end
        end
    end
end
end
```

はじめに～プロファイルを掛けよう

- どこがボトルネックになっているのかプロファイラーで把握しましょう
- プロファイラーの開き方
 - 「ホーム」タブの「実行および時間の計測」



- 「アプリ」タブの「プロファイラー」

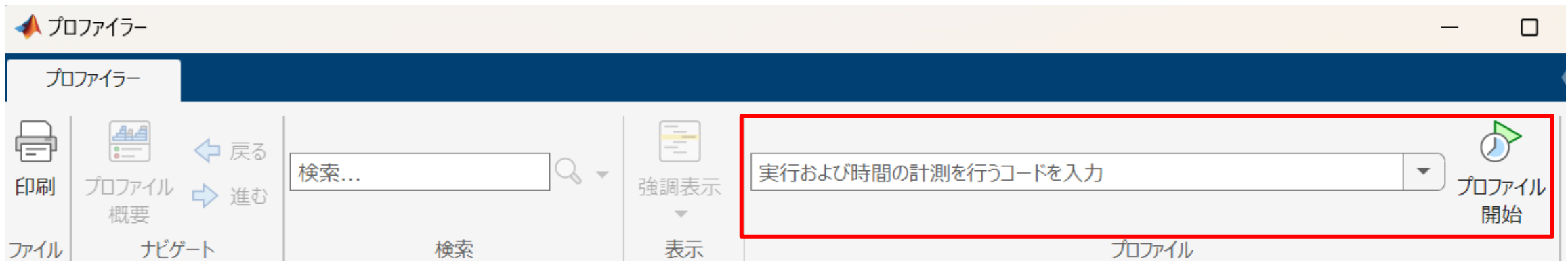


はじめに～プロフィールを掛けよう

- スクリプト名を入力して「プロフィール開始」をクリック

Or

- 「プロフィール開始」をクリックしてからいつも通りスクリプトを実行



はじめに～プロファイルを掛けよう

プロファイラー

プロファイル

検索...

airlineBefore

実行および時間の計測

プロファイル概要 (合計時間: 199.750 秒)

フレームグラフ

tabular.dotParenReference

airlineBefore

Profile Summary

パフォーマンス 時間を使用して 12-5-2025 15:04:02 が生成されました。

関数名	呼び出し	合計時間 (s)	自己時間* (s)	合計時間プロット (自己時間 / 合計時間)
airlineBefore	1	199.742	50.305	
tabular.dotParenReference	34092348	147.282	147.282	

リンクをクリック

▼ 最も時間を要する行

行番号	コード	呼び出し	合計時間 (s)	% 時間	時間プロット
7	<code>if airlineData.Year(n) == y</code>	32610072	186.112	93.2%	
8	<code>if airlineData.Month(n) == m</code>	1482276	8.647	4.3%	
1	<code>airlineData = readtable("airlinesmall.csv");</code>	1	2.168	1.1%	
12	<code>end</code>	32610072	1.268	0.6%	
11	<code>end</code>	32610072	1.098	0.5%	
他のすべての行			0.448	0.2%	
合計			199.742	100%	

200 秒のうち、最初のif 分の処理が186 秒 (約93%)を占めている

コードアナライザーのヒントを参考にする

- コードアナライザーのヒントを参考にコードを修正

```
code_analyzer_demo.m
1 % NG
2 for n=1:1000
3     a(n) = rand;
4 end
5
6 hist(a)
```

警告がオレンジ
で表示されます

マウスカーソルを当て
ると警告内容が表示さ
れます

ヒントに従って
修正します



```
code_analyzer_demo.m
1 % OK
2 a = zeros(1, 1000);
3 for n=1:1000
4     a(n) = rand;
5 end
6
7 histogram(a)
```

a(n) = rand;

⚠ 変数 'a' のサイズが (スクリプト内の) ループの反復ごとに変更されているようです。高速化するために、事前割り当てを検討してください。 詳細 ▾

hist(a)

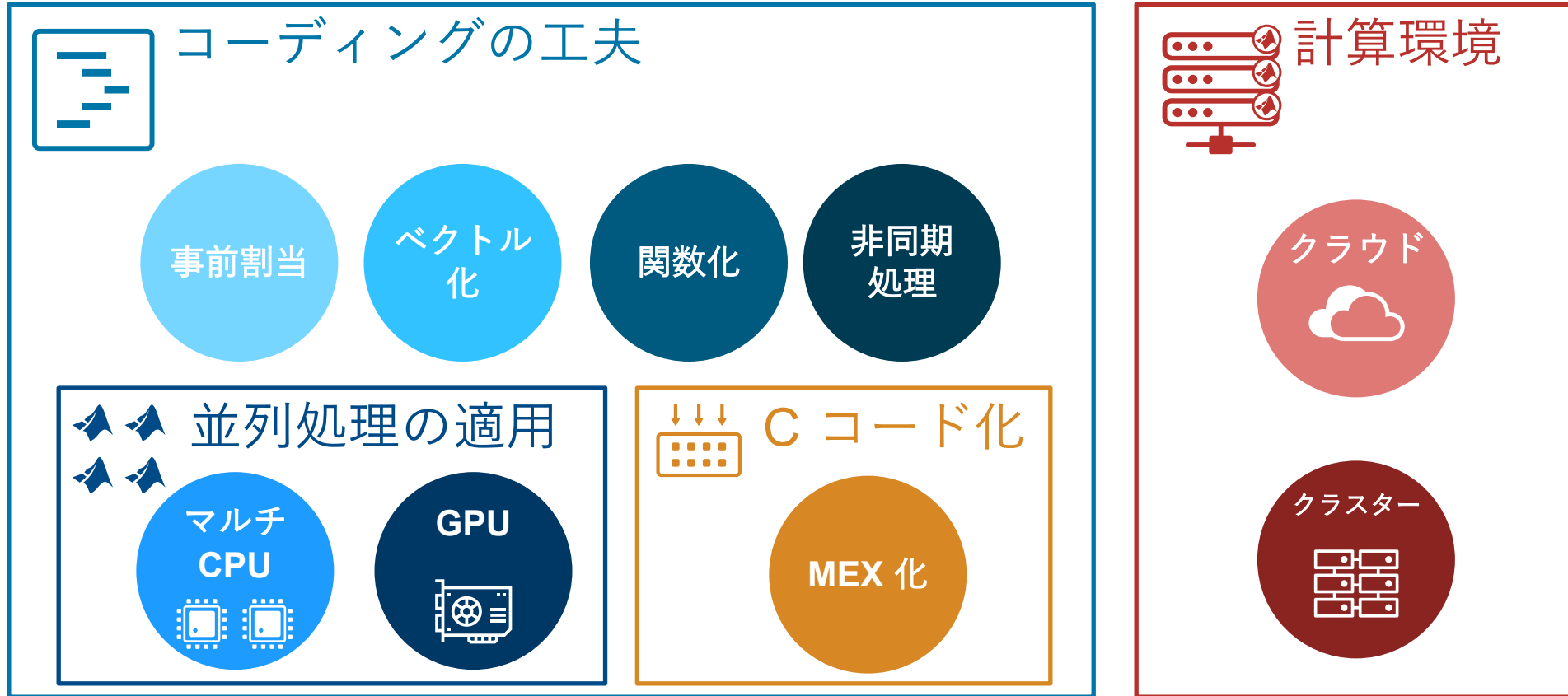
⚠ 'hist' は推奨されていません。代わりに 'histogram' を使用してください。

高速化のためのコードの書き方

～高速化の手段を検討する～



高速化の手段

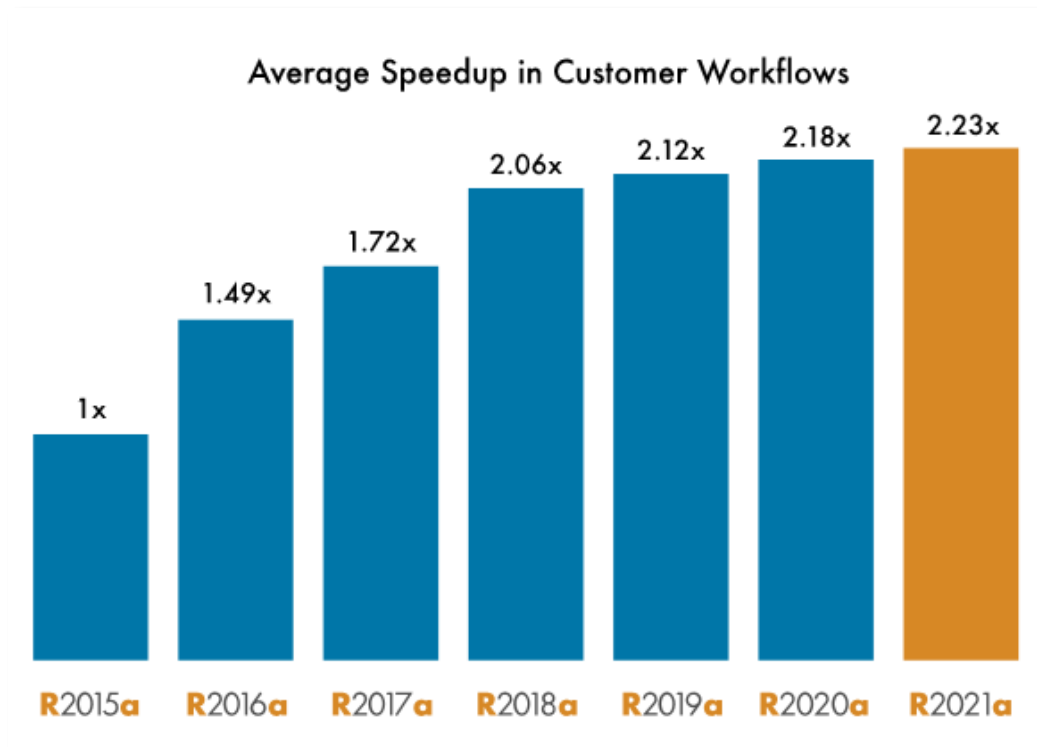


【ドキュメント】パフォーマンス向上の手法

https://jp.mathworks.com/help/matlab/matlab_prog/techniques-for-improving-performance.html

バージョンアップをする

- MATLAB はバージョンアップする毎にパフォーマンスが改善されています



▼ R2024a

New Features, Bug Fixes, Compatibility Considerations

R2024a: バグ修正

Performance

- › `datetime` Format Parsing: Improved performance when parsing common numeric formats
- › `imread` and `imwrite` Functions: Improved performance with JPEG files
- › `smoothdata2` and `fillmissing2` Functions: Improved performance for moving average methods
- › `mldivide` Function: Improved performance with tridiagonal matrices
- › `audioplayer` Function : Improved performance with longer duration audio signals
- › Files Panel in MATLAB Online: Improved performance when previewing large MAT-files
- › Workspace Panel in MATLAB Online: Improved performance when loading large number of variables
- › Plots in Apps and MATLAB Online: Ticks and tick labels update faster when zooming
- › Plots in Apps and MATLAB Online: Improved responsiveness when interacting with partially transparent 3-D plots

<https://jp.mathworks.com/help/matlab/release-notes.html?category=performance>

PC単体でもできるTips

MATLAB の標準機能でできる高速化の様々なテクニック

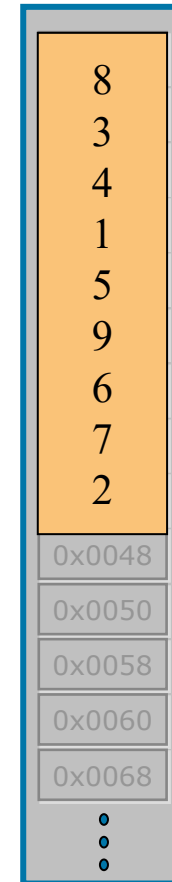


列優先アクセス

```
>> x = magic(3)
```

```
x =
```

8	1	6
3	5	7
4	9	2



列優先

2次元またはそれ以上の行列の操作を行う場合、一般的に列方向で行列の要素にアクセスするほうが高速

<https://jp.mathworks.com/matlabcentral/answers/454074->

MATLAB 配列のデータ格納

参考：[行列の操作において、行方向優先と列方向優先のどちらが速いですか？](#)

列優先アクセス

行方向で書いたコード

```
tic
n = 2e4;
x = randn(n);
y = zeros(n);
for r = 1:n
    for c = 1:n
        if x(r, c) >= 0
            y(r, c) = x(r, c);
        end
    end
end
t1 = toc
```

11.7 秒

列方向で書いたコード

```
tic
n = 2e4;
x = randn(n);
y = zeros(n);
for c = 1:n
    for r = 1:n
        if x(r, c) >= 0
            y(r, c) = x(r, c);
        end
    end
end
t2 = toc
```

4.8 秒

配列の事前割り当て

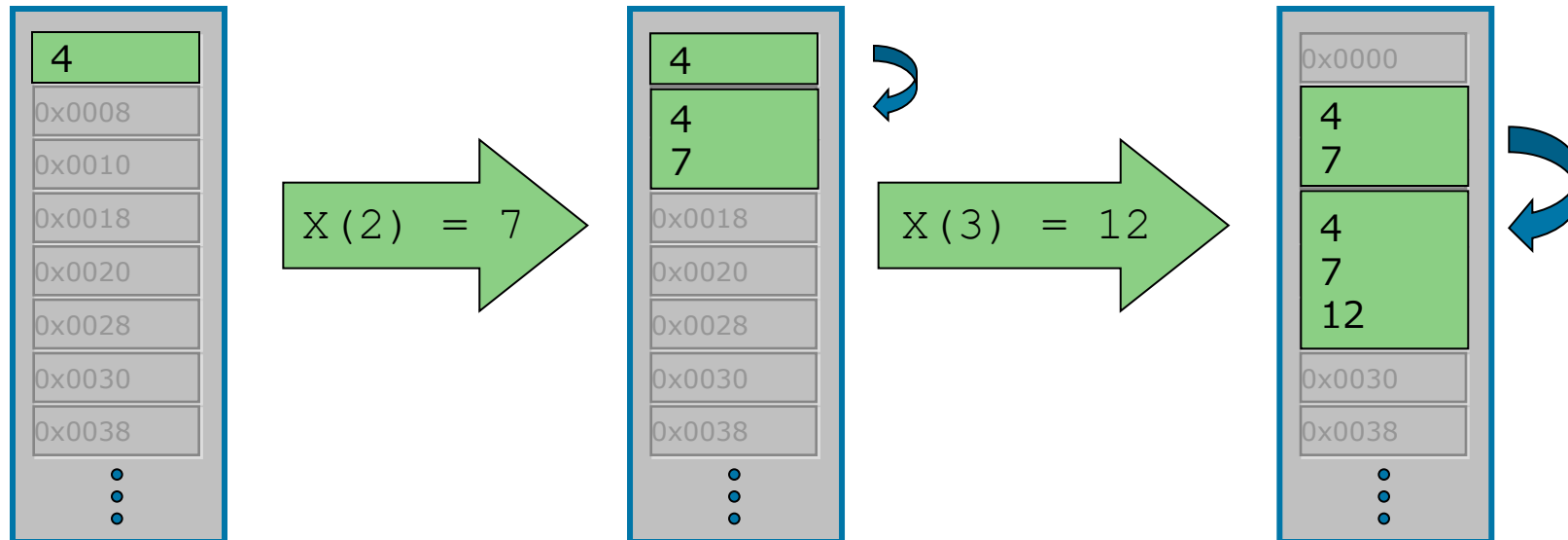
事前割り当てなしの場合

```
>> x = 4
```

```
>> x(2) = 7
```

```
>> x(3) = 12
```

配列のリサイズは
コストがかかる！



配列の事前割り当て

事前割り当てありの場合

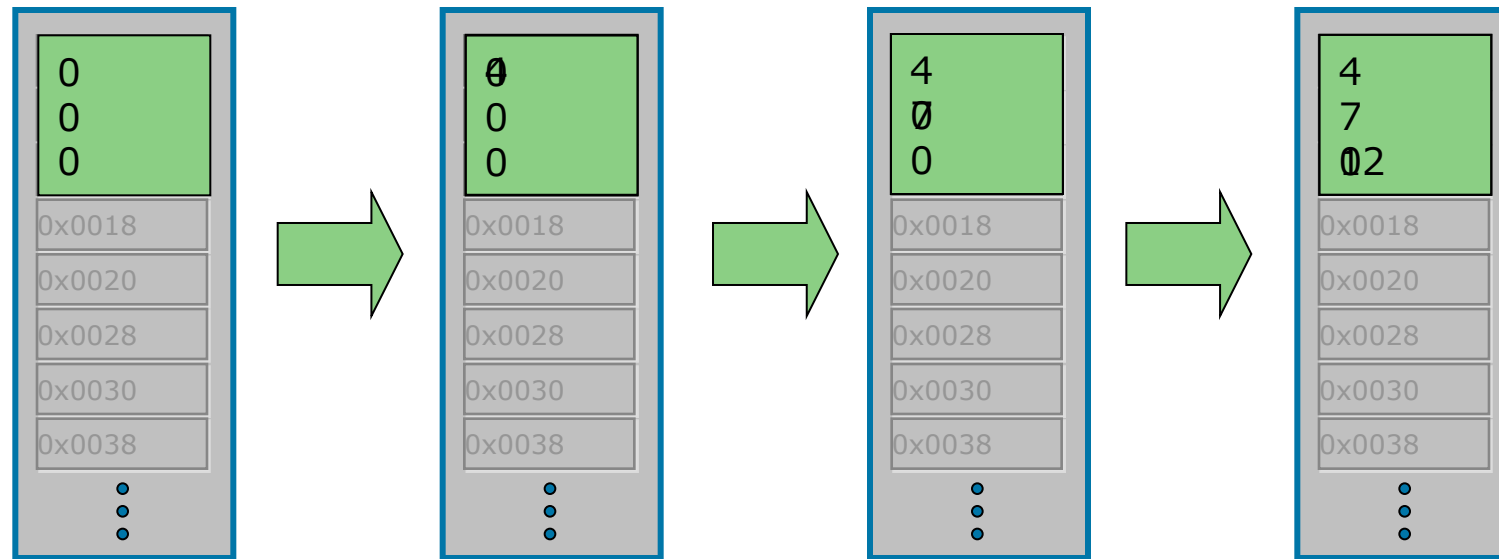
```
>> x = zeros(3,1)
```

```
>> x(1) = 4
```

```
>> x(2) = 7
```

```
>> x(3) = 12
```

無駄なコピーが
発生しない！



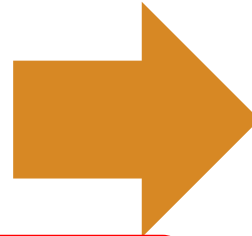
配列の事前割り当て

- 事前に配列のサイズを指定しておく

```
tic
for n=1:100000000
    a(n) = 1;
    b(n) = 2;
end
toc
```

配列のサイズが動的に変更され、
オーバーヘッドが大きい

経過時間は 17.841129 秒です。



事前割当

```
tic
a = zeros(100000000, 1);
b = zeros(100000000, 1);
for n=1:100000000
    a(n) = 1;
    b(n) = 2;
end
toc
```

経過時間は 0.334331 秒です。

なるべく組み込み関数を利用する

```
tic
a = zeros(100000000, 1);
b = zeros(100000000, 1);
for n=1:100000000
    a(n) = 1;
    b(n) = 2;
end
toc
```

経過時間は 0.334331 秒です。



組込関数の利用

```
tic
a = ones(100000000, 1);
b = 2*ones(100000000, 1);
toc
```

経過時間は 0.284328 秒です。

```
s = 0;
A = 1:100000000;
tic
for n=1:length(A)
    s = s + A(n);
end
toc
```

経過時間は 0.184148 秒です。



```
A = 1:100000000;
tic
s = sum(A);
toc
```

組込関数の利用

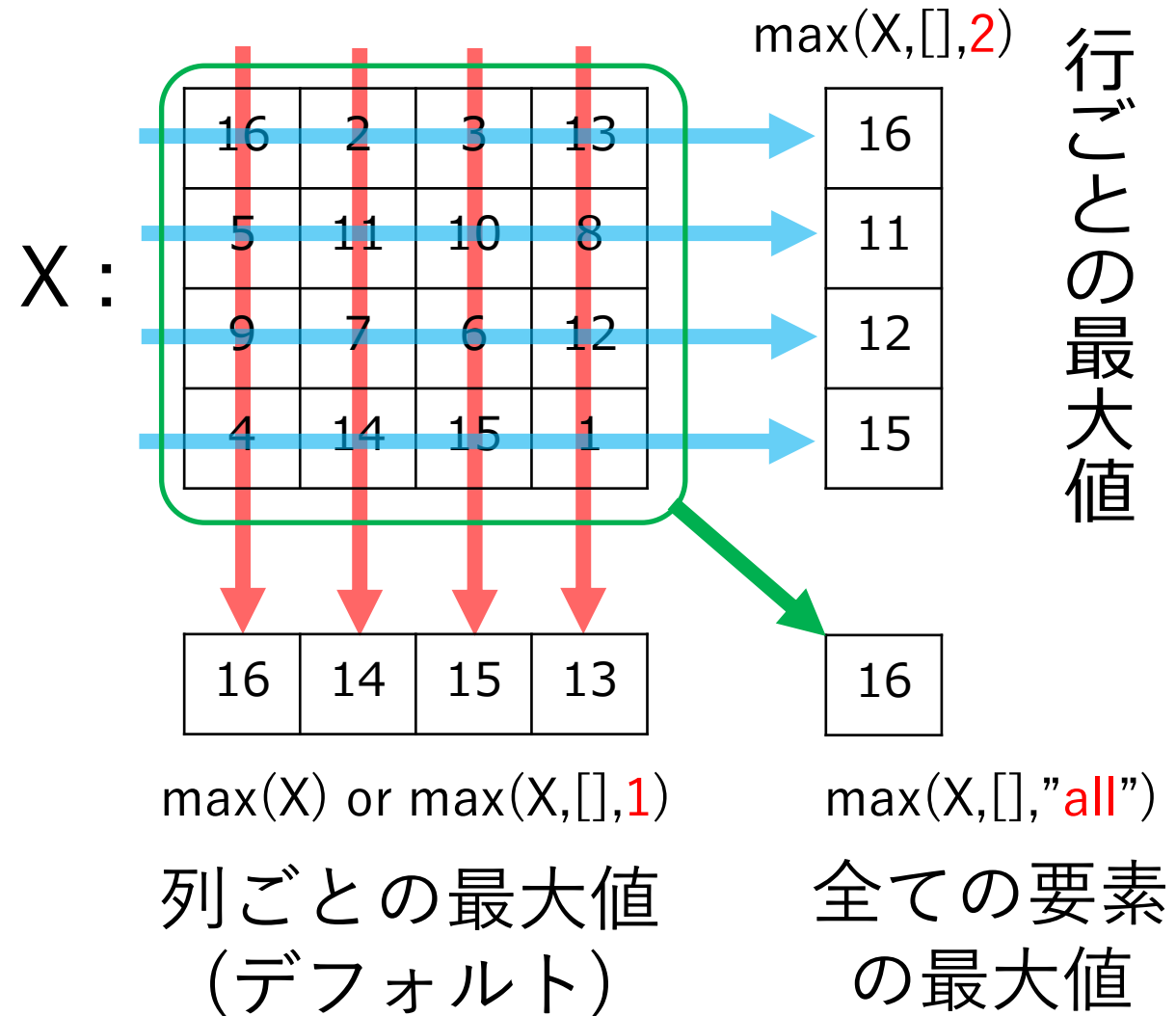
経過時間は 0.039842 秒です。

記述例	生成される行列
zeros(3, 4)	要素が全て 0 の 3 行 4 列の行列
zeros(3)	要素が全て 0 の 3 行 3 列の行列
eye(5)	5 行 5 列の単位行列
rand(3, 4)	区間 (0 1) の一様乱数からなる 3 行 4 列の行列
randn(3, 4)	平均 0 分散 1 の正規乱数からなる 3 行 4 列の行列
[A B]	行列 A と行列 B を横連結した行列
[A; B]	行列 A と行列 B を縦連結した行列

関数	記号
最大値	max
最小値	min
総和	sum
平均	mean
中央値	median
分散	var
ソート	sort
FFT	fft

組込関数における処理の次元方向

関数	記号
最大値	max
最小値	min
総和	sum
平均	mean
中央値	median
分散	var
ソート	sort
FFT	fft



インデックス処理

論理配列：論理値（0 または 1）からなる配列

x :

1	2	3	4	5	3	2	1
---	---	---	---	---	---	---	---

| = $x < 3$



| :

1	1	0	0	0	0	1	1
---	---	---	---	---	---	---	---

例題：xの要素の内、3より小さいものの**個数**を求める

`sum(x < 3)`

例題：xの要素の内、3より小さいものの**位置**を求める

`find(x < 3)`

1	2	7	8
---	---	---	---

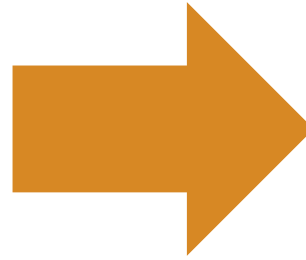
インデックス処理

- インデックス処理で条件処理をまとめて行う

```
x = randi(5, 10000, 10000);  
tic  
for n=1:10000  
    for m=1:10000  
        if x(n, m) == 1  
            x(n, m) = 0;  
        end  
    end  
end  
toc
```

for 文や if 文

経過時間は 1.076086 秒です。



```
x = randi(5, 10000, 10000);  
tic  
x(x==1) = 0;  
toc
```

インデックス処理

経過時間は 0.388458 秒です。

for 文の代わりに cellfun、arrayfun、structfun を使う方法も

<https://jp.mathworks.com/matlabcentral/answers/103853>

ベクトル化 ~ for ループを使わない! ~

ベクトル化の利点

- コードが数式のように記述され、コードが理解しやすい
- コードが短くなり、バグが発生し難くなる
- 高速に実行できる(最適化された演算が実行される: スパコンも同様)

ループによる逐次処理 (時間処理) を空間処理で置き換える

例 実数の内積計算

$$z = \sum_{i=1}^n x_i y_i = \mathbf{x}^T \mathbf{y}$$

ベクトル化しない場合

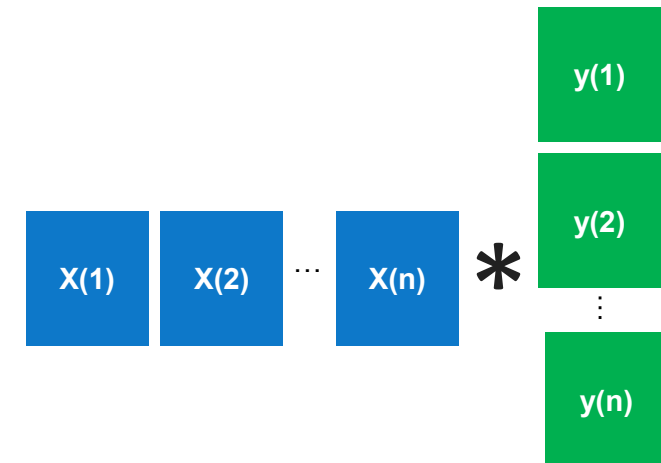
```
tic
for i = 1:1000
    %一つずつ取り出して計算
    z = z+x(i)*y(i);
end
toc
```

経過時間は 0.007683 秒です。

ベクトル化した場合

```
tic
%ベクトルのまま計算
zz = x'*y;
toc
```

経過時間は 0.001438 秒です。



高速化!

インデックスによるモンテカルロ法

■ モンテカルロ法による円周率の計算

正方形の中にランダム点を生成

$$M : N \doteq \frac{\pi}{4} : 1$$

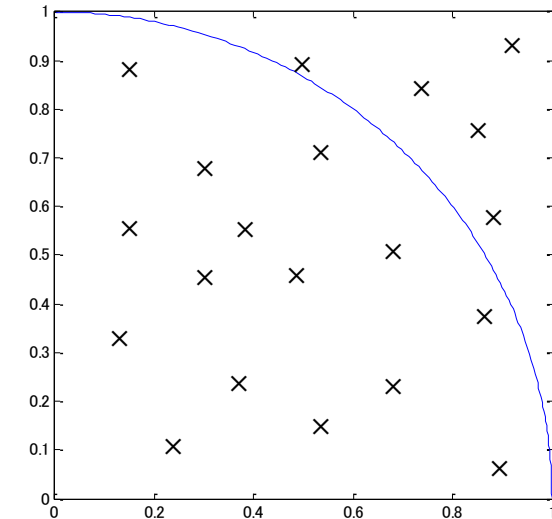
点の個数の比

面積の比

$$\pi \doteq \frac{4M}{N}$$

M : 四分円の内側に入った点の個数

N : 乱数で発生させた点の総数



for ループを使用

```
clear
rng('default')
N = 1e7;

tic
M = 0;
for k = 1:N
    x = rand(1); y = rand(1);
    r2 = x^2 + y^2;
    if r2 < 1
        M = M + 1;
    end
end
toc

pi_hat = 4 * M / N;
disp(pi_hat)
```

経過時間は 1.040366 秒です。

3.1415

ベクトル化

```
clear
rng('default')
N = 1e7;

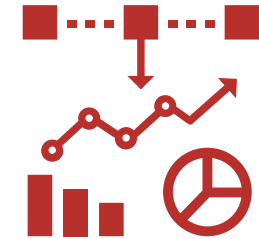
tic
xy = rand(N,2);
ind = sum(xy.^2,2) < 1;
M = sum(ind);
toc

pi_hat = 4 * M / N;
disp(pi_hat)
```

経過時間は 0.186148 秒です。

3.1418

可視化の高速化

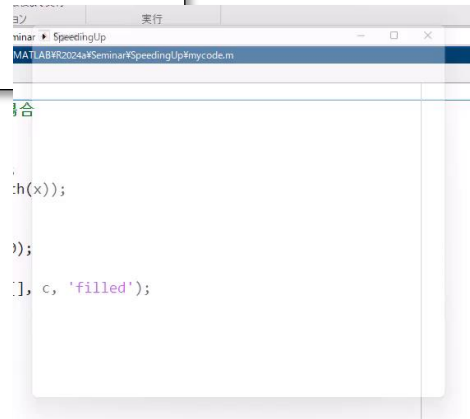


- プロットを繰り返す場合は、プロットを毎回実行させるのではなく、中のデータ (XData、YData など)を更新する方法で高速に実行できます

毎回scatter コマンドを実行

```
rng("default")
x = linspace(0, 3*pi, 200);
c = linspace(1, 3*pi, length(x));
tic
for n=1:100
    y = cos(x) + rand(1,200);
    scatter(x, y, [], c, 'filled')
    drawnow limitrate
end
t1 = toc
```

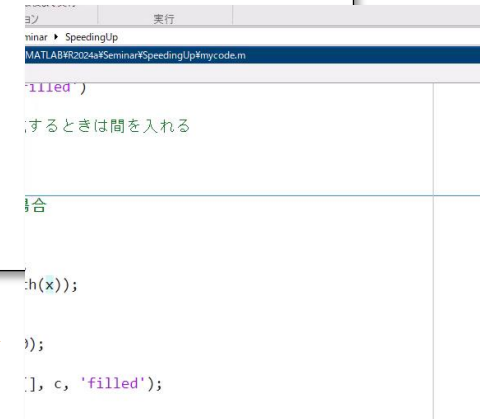
1.25 秒



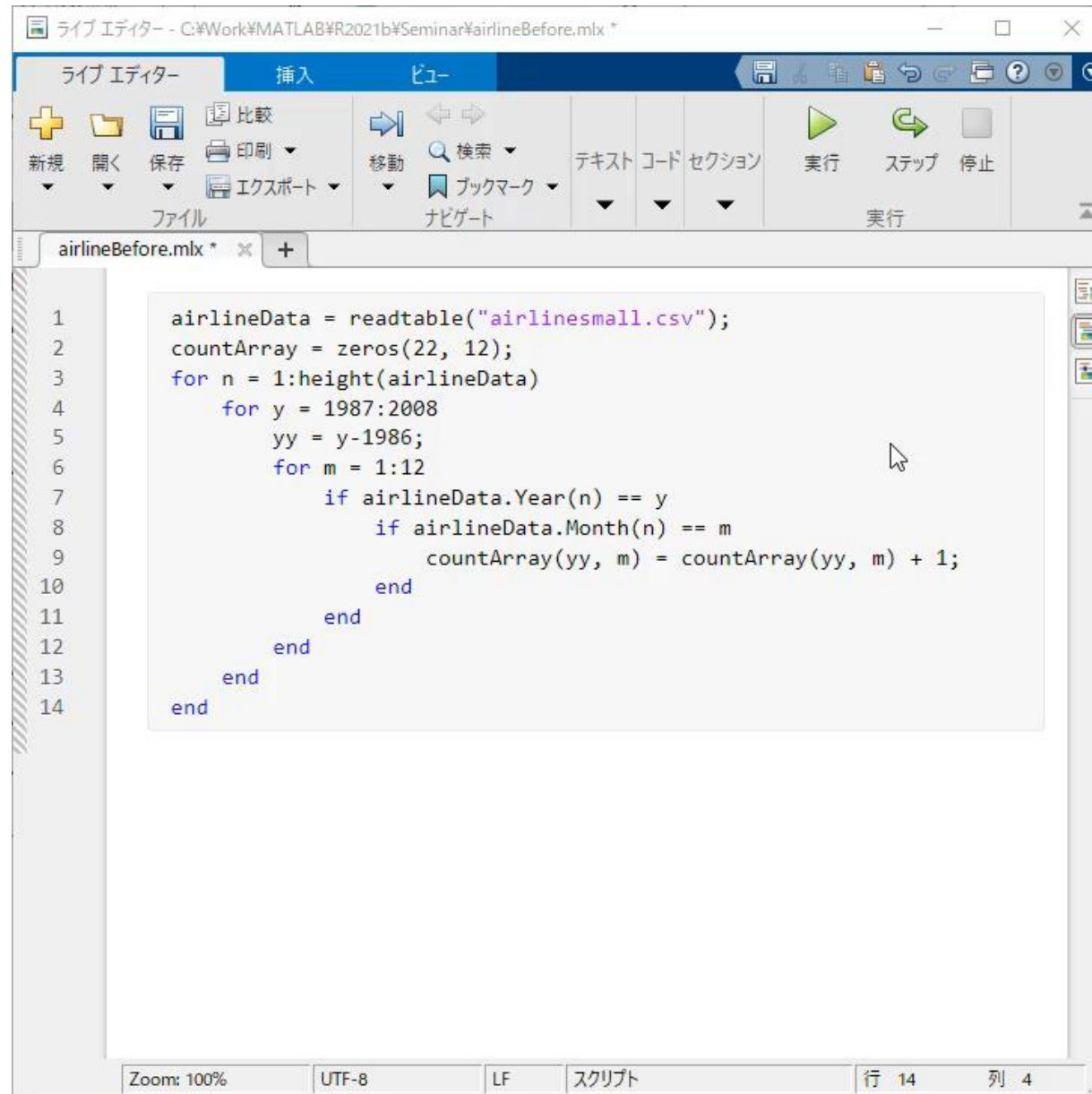
2 回目以降はYData を更新

```
rng("default")
x = linspace(0, 3*pi, 200);
c = linspace(1, 3*pi, length(x));
tic
for n=1:100
    y = cos(x) + rand(1,200);
    if n == 1
        s = scatter(x, y, [], c, 'filled');
    else
        s.YData = y;
        drawnow limitrate
    end
end
t2 = toc
```

0.38 秒



高速化の適用



```
1 airlineData = readtable("airlinesmall.csv");
2 countArray = zeros(22, 12);
3 for n = 1:height(airlineData)
4     for y = 1987:2008
5         yy = y-1986;
6         for m = 1:12
7             if airlineData.Year(n) == y
8                 if airlineData.Month(n) == m
9                     countArray(yy, m) = countArray(yy, m) + 1;
10                end
11            end
12        end
13    end
14 end
```

Zoom: 100% UTF-8 LF スクリプト 行 14 列 4

プロファイラで効果測定

```

airlineData = readtable("airlinesmall.csv");
countArray = zeros(22, 12);
for n=1:height(airlineData)
    for y = 1987:2008
        yy = y-1986;
        for m = 1:12
            if airlineData.Year(n) == y
                if airlineData.Month(n) == m
                    countArray(yy, m) = countArray(yy, m) + 1;
                end
            end
        end
    end
end
end

```

変更前

約200 秒

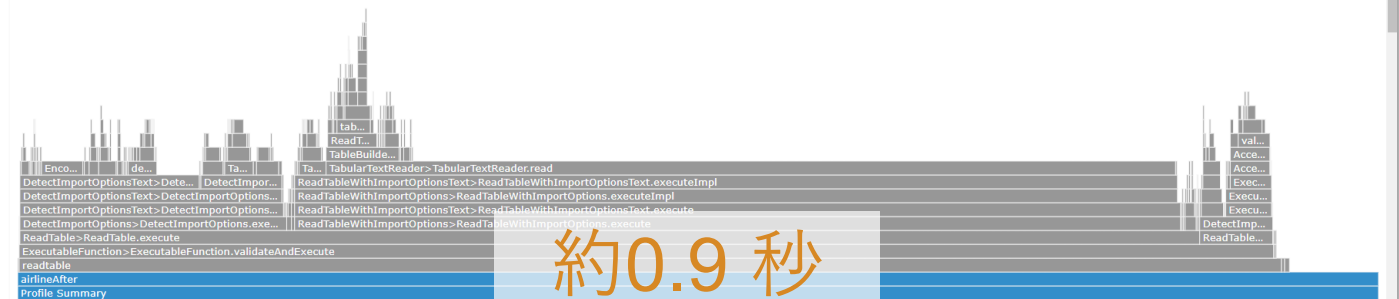
```

airlineData = readtable("airlinesmall.csv");
countArray = zeros(22, 12);
for y = 1987:2008
    yy = y-1986
    idx1 = airlineData.Year == y;
    for m = 1:12
        idx = idx1 & airlineData.Month == m;
        countArray(yy, m) =sum(idx);
    end
end

```

変更後

フレームグラフ



約0.9 秒

パフォーマンス 時間を 12-5-2025 15:10:19 が生成されました。

関数名	呼び出し	合計時間 (s)	自己時間* (s)	合計時間プロット (自己時間 / 合計時間)
airlineAfter	1	0.857	0.056	
readtable	1	0.795	0.004	
ExecutableFunction>ExecutableFunction.validateAndExecute	1	0.790	0.001	
ReadTable>ReadTable.execute	1	0.743	0.002	
ReadTableWithImportOptions>ReadTableWithImportOptions.execute	1	0.558	0.000	
ReadTableWithImportOptionsText>ReadTableWithImportOptionsText.execute	1	0.557	0.000	
ReadTableWithImportOptionsText>ReadTableWithImportOptionsText.executeImpl	1	0.557	0.000	
ReadTableWithImportOptionsText>ReadTableWithImportOptionsText.executeImpl	1	0.557	0.003	
TabularTextReader>TabularTextReader.read	1	0.535	0.484	
DetectImportOptions>DetectImportOptions.execute	1	0.168	0.001	

ここまでのまとめ

高速化テクニック	参考情報
プロファイリングによる分析	https://jp.mathworks.com/help/matlab/matlab_prog/profiling-for-improving-performance.html
データへの列優先アクセス	https://jp.mathworks.com/matlabcentral/answers/454074-
配列の事前割当	https://jp.mathworks.com/help/matlab/matlab_prog/preallocating-arrays.html
組み込み関数の利用	https://www.mathworks.com/content/dam/matworks/fact-sheet/jp-matlab-basic-functions-reference.pdf
インデックス処理	https://jp.mathworks.com/help/matlab/math/array-indexing.html
ベクトル化	https://jp.mathworks.com/help/matlab/matlab_prog/vectorization.html

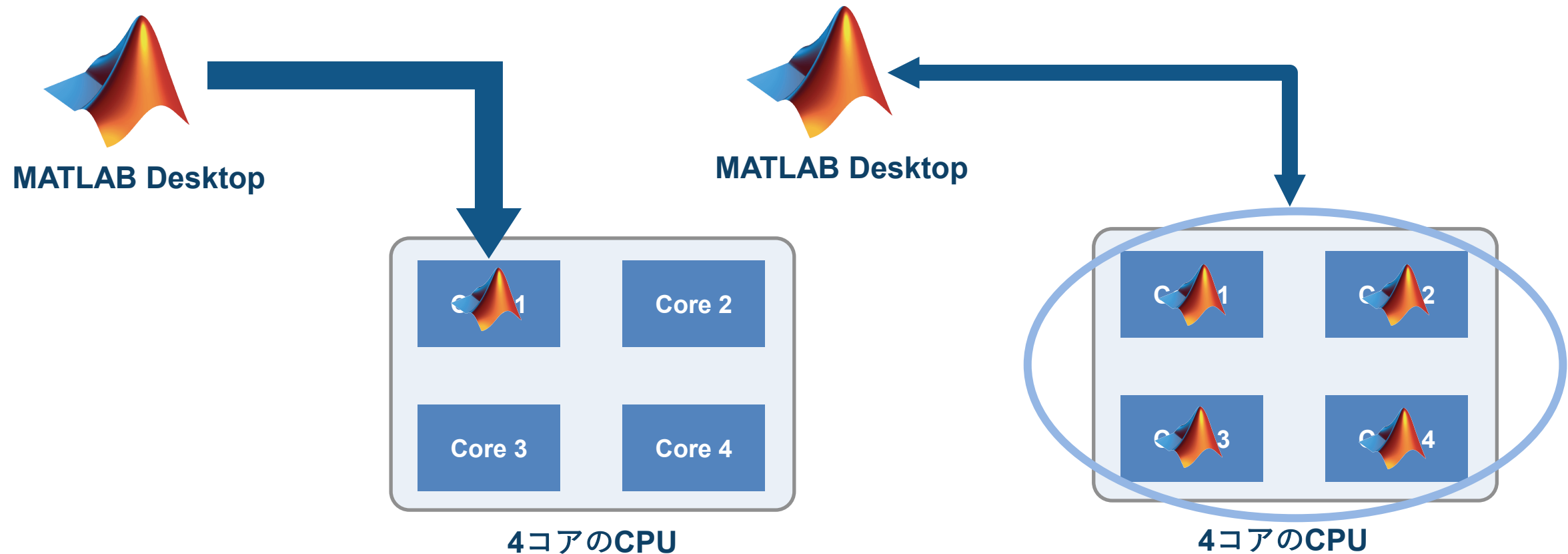
PC単体でもできるTips

並列処理

マルチコアの活用

Parallel Computing Toolboxを使った並列処理

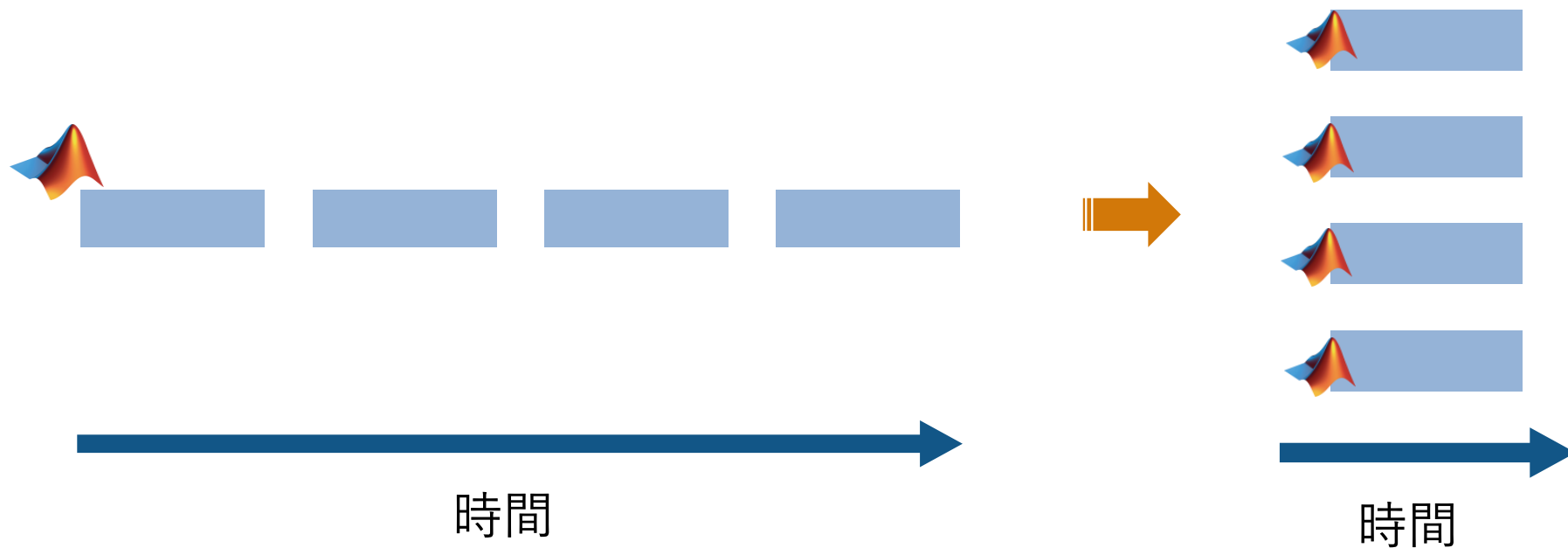
通常MATLABコードは一つのコアの上で動作※



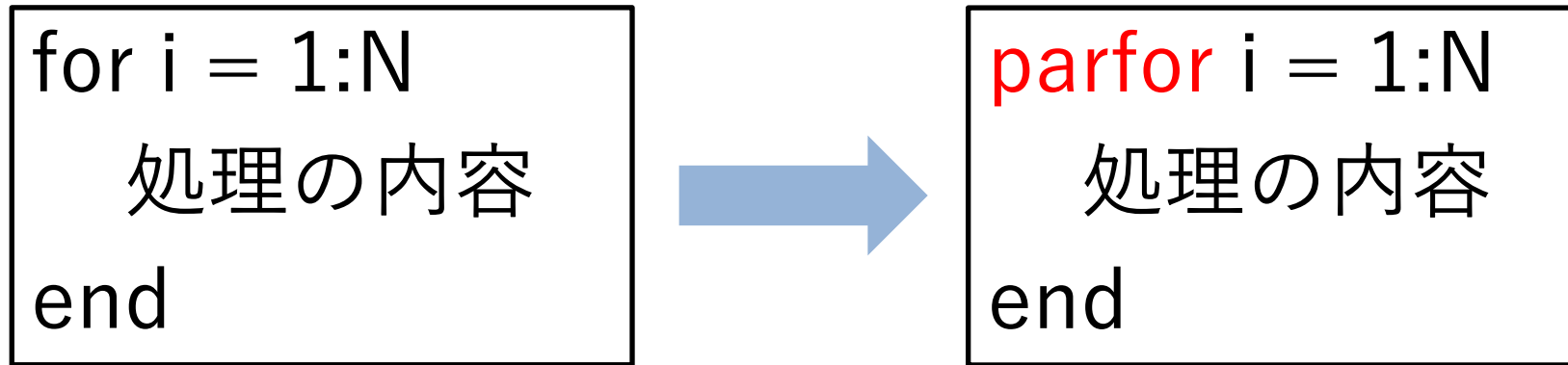
※ 2008年以降、fft, eig, svd, sort等、多数の線形代数および数値計計算の関数はマルチスレッドで動作

並列 for ループ (parfor)

- 条件: タスク同士に依存関係や通信が発生しない
- 例: パラメータスイープ、モンテカルロシミュレーション



基本的なparforの実装方法→「for」を「parfor」に置き換えるだけ



※Parallel Computing Toolboxが必要です

※初回実行時は並列プールを起動するため処理に時間がかかります
→ループ内処理の内容によっては遅くなる可能性あり

ドキュメント: [parfor](#)

parfor の注意点

- 各 **for** ループは独立実行
- ループの実行順序はランダム
- 複数の **for** ループがある場合
 - 外側の **for** ループを **parfor** に変更
 - 2 つ以上の **parfor** は使用不可

```
parfor I = 2:10
```

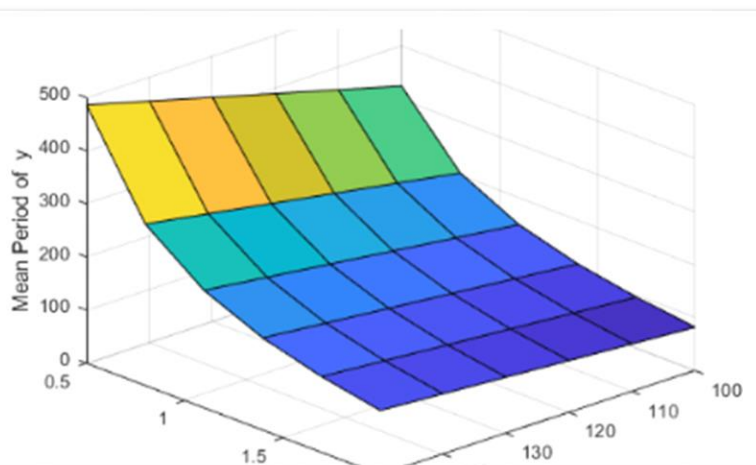
❌ The PARFOR loop cannot run due to the way variable 'A' is used. Details ▾

```
end
```

	1回目	2回目
parfor I=1:5	4	2
disp(I)	3	1
	2	5
end	1	4
	5	3

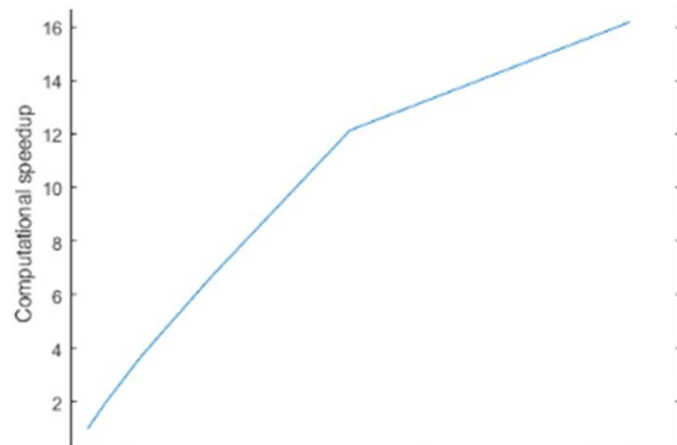
```
A = zeros(4,5);  
parfor j = 1:4  
    for k = 1:5  
        A(j,k) = j + k;  
    end  
end
```

parforを用いた例題



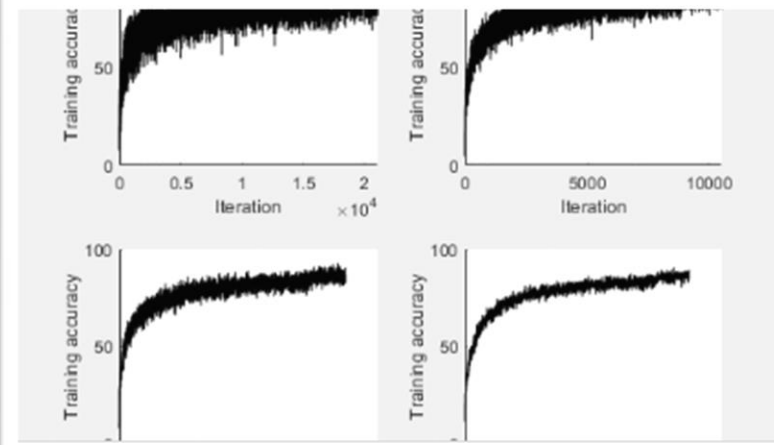
parfor を使用したパラメーター スイープ中のプロット

パラメーター スイープを並列実行して、並列計算中に進行状況をプロットします。



parfor を使用したモンテカルロ コードの高速化

この例では、parfor ループを使用してモンテカルロ コードを高速化する方法を説明します。モンテカルロ法は、物理学、数学、生物学、金融…



parfor を使用した複数の深層学習 ネットワークの学習

この例では、parfor ループを使用して、学習オプションについてのパラメーター スイープを実行する方法を説明します。
(Deep Learning Toolbox)

PC単体でもできるTips

GPU、スーパーコンピュータとの連携

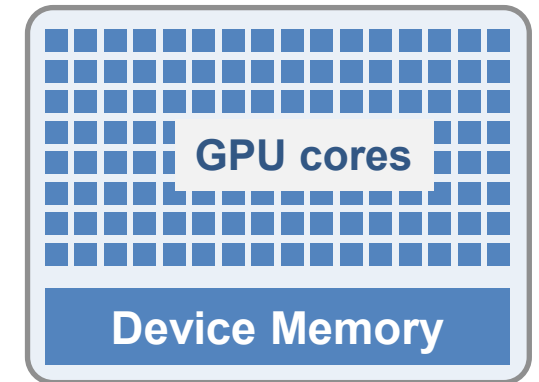
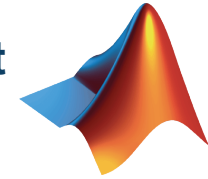
GPU (Graphics Processing Unit) とは

グラフィックスカード/ビデオカード上のプロセッサ

- 本来はグラフィックス処理向け
- 科学技術計算にも応用(GP-GPU)
- 整数および浮動小数点数計算向けの超並列プロセッサ
 - 1枚のカードに数百のコアを搭載
 - CPUに比べて汎用性に関して劣る
- 高速な専用メモリを搭載
- Parallel Computing Toolboxにより、NVIDIA GPUを使用可能(500以上の組み込み関数に対応)



MATLAB client
or Worker



例: 2次元フーリエ変換

GPU計算 – データ転送関数の利用

ドキュメントは [こちら](#)

- 5000 × 5000 の 2 次元データに対するフーリエ変換

```
#include "mex.h"

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

/*****

/* MATLAB stores complex numbers in separate arrays for the real and
imaginary parts. The following functions take the data in
this format and pack it into a complex work array, or
unpack it, respectively.
We are using cufftComplex defined in cufft.h to handle complex on Windows and Linux

*/

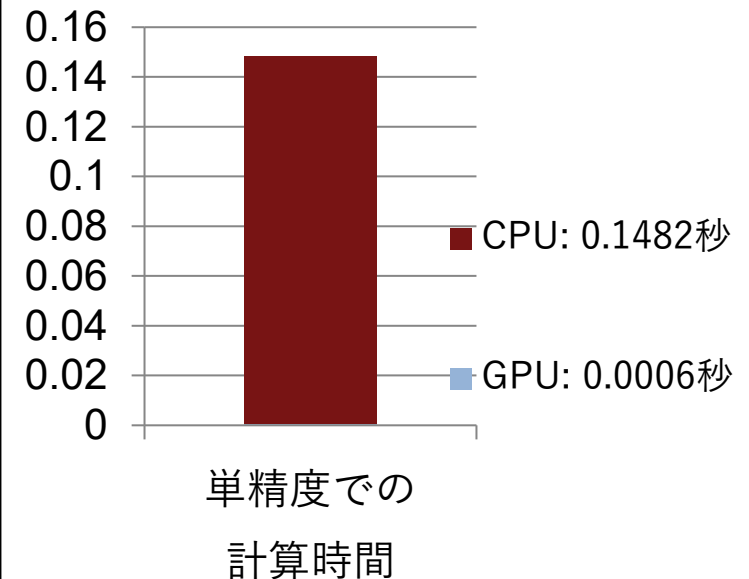
void pack_r2c(cufftComplex *output_float,
double *input_re,
int Ntot)
{
    int i;
    for (i = 0; i < Ntot; i++)
    {
        output_float[i].x = input_re[i];
        output_float[i].y = 0.0f;
    }
}

void pack_c2c(cufftComplex *output_float,
double *input_re,
double *input_im,
int Ntot)
{
    int i;
    for (i = 0; i < Ntot; i++)
    {
        output_float[i].x = input_re[i];
        output_float[i].y = input_im[i];
    }
}
```

PCTで等価なコード

C言語 + CUDA+MEX
コメント含めて200行以上

```
>> GX = gpuArray(X);
>> GY = fft2(GX);
>> Y = gather(GY);
```



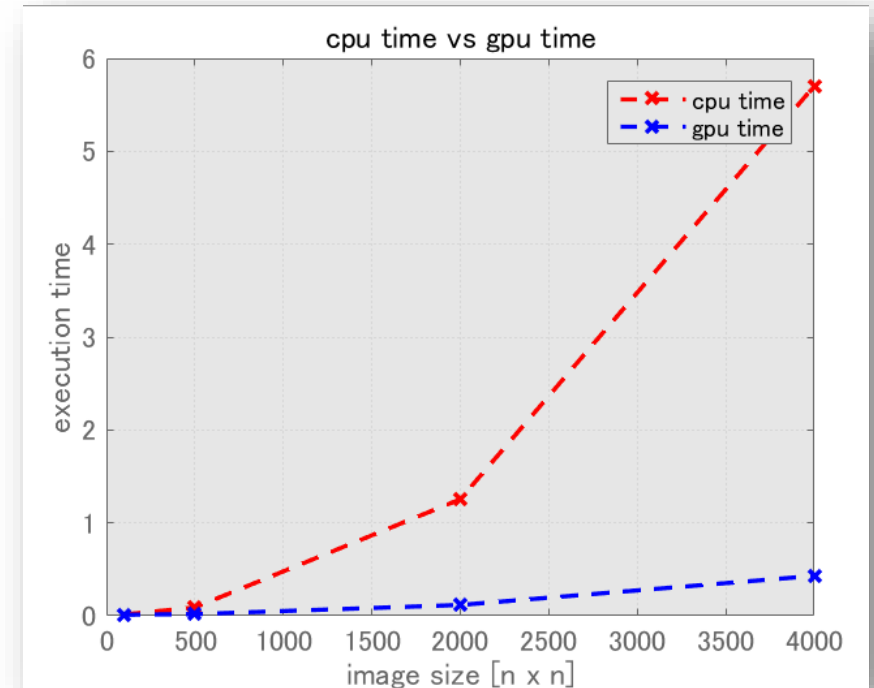
例: 画像のフィルタリング

GPU計算 – データ転送関数の利用

- 写真にフィルタをかけて、キャンバス効果をつける

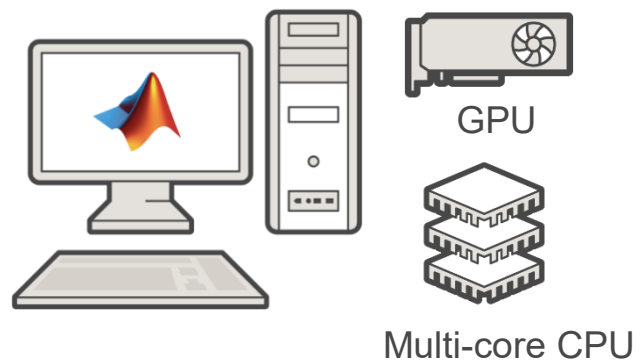


```
% GPU にデータ転送  
imGPU = gpuArray(im);  
  
% キャンバス効果をつける  
canvasGPU = canvasEffect(imGPU);  
  
% GPU からデータ回収  
canvas = gather(canvasGPU);
```



スーパーコンピュータとの連携

スーパーコンピュータとの連携



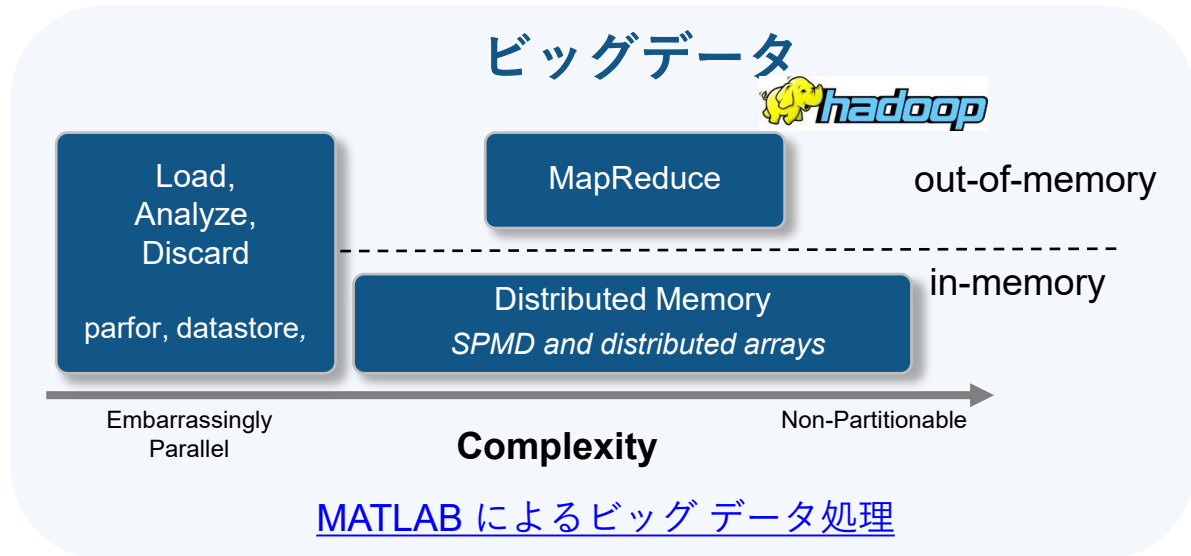
Parallel Computing Toolbox

スケーラブルな展開

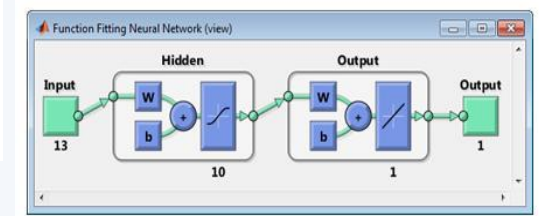
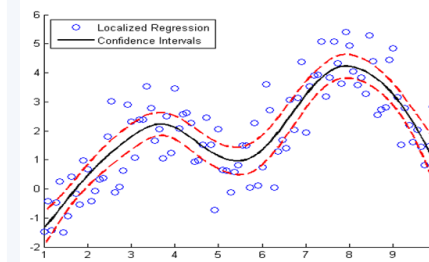


MATLAB Parallel Server

並列計算が活用できる分野の例

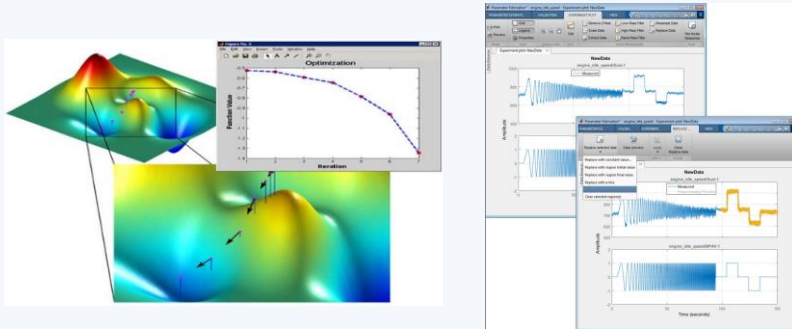


統計解析、機械学習



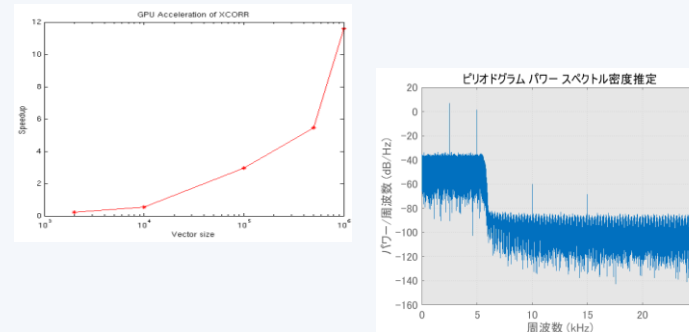
統計計算の速度の向上
複数の深層学習ネットワークの学習

最適化



Optimization Toolbox 関数の並列計算

信号処理



GPUを使用した相関の高速化

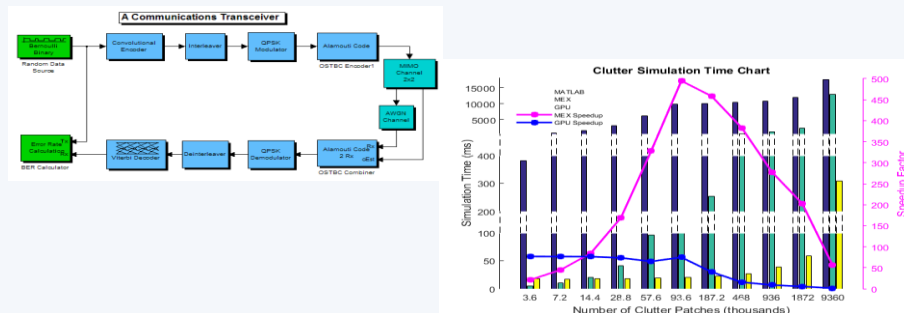
画像処理



GPUでの画像処理
大規模なイメージセットの処理

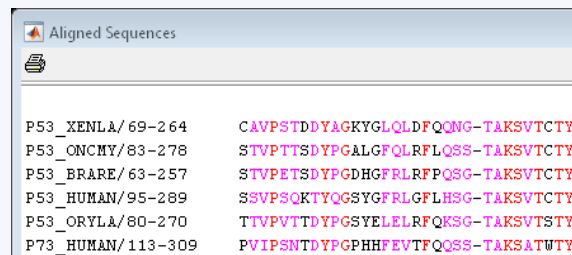
並列計算が活用できる分野の例

通信、コミュニケーション



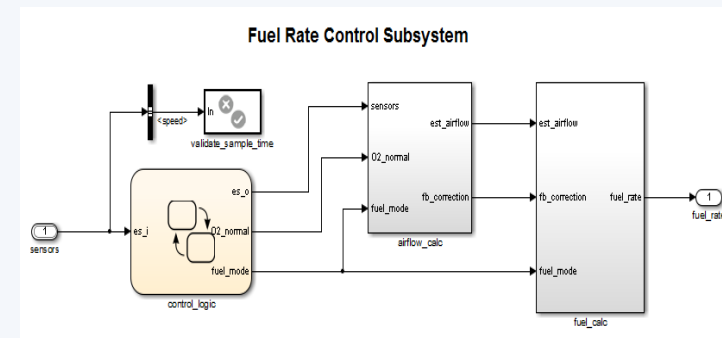
GPU を使ったシミュレーションの高速化

計算生物学



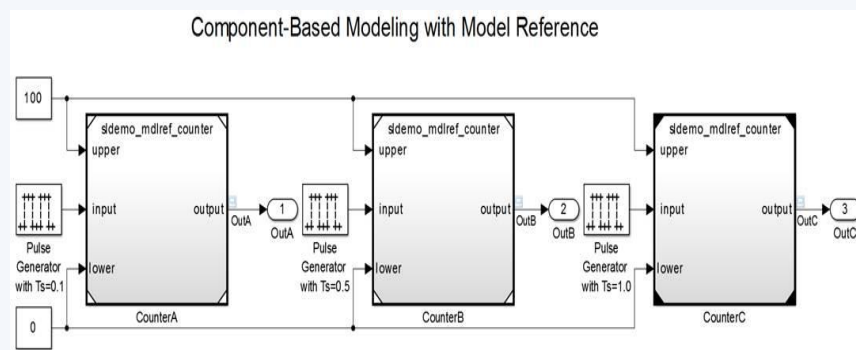
大規模質量分析信号の前処理

Simulink



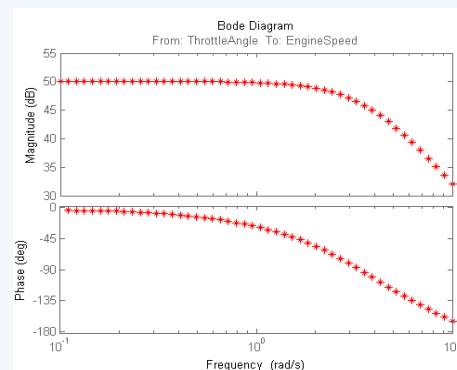
並列シミュレーションの実行
ラピッドアクセラレータモードでのパラメタスイープ

コード生成



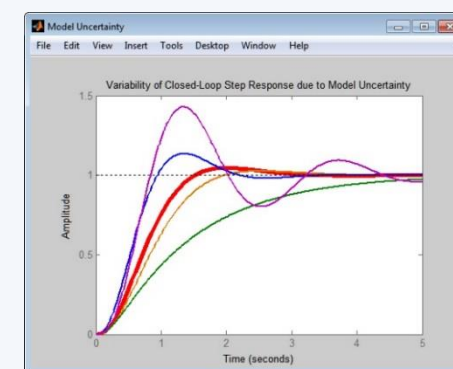
コード生成とビルドの並列化

制御設計



並列計算を使用した周波数応答推定の高速化

ロバスト制御



並列計算を使用したチューニングの高速化

まとめ

1. MATLAB概要
2. 高速化のためのコードの書き方
 - PC単体でもできるTips
 - プロファイラーの活用
 - 列優先アクセス
 - 配列の事前割り当て
 - ベクトル化
 - グラフの描画
 - マルチコアの活用（並列処理）
 - GPU、スーパーコンピュータとの連携
 - GPU
 - スーパーコンピュータとの連携





© 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.

基本的な使い方から専門分野まで学べる MATLAB eラーニング教材

matlabacademy.mathworks.com/jp



入門コース- どなたでも無料でアクセスいただけます



MATLABを使い始める

-  MATLAB 入門
-  ディープラーニング入門
-  機械学習入門
-  画像処理入門
-  信号処理入門
-  Statistics Onramp

-  アプリ作成入門
-  最適化入門
-  コンピュータービジョン入門
-  強化学習入門
-  オブジェクト指向プログラミング入門
-  無線通信入門

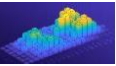



Simulinkを使い始める

-  Simulink 入門
-  Simscape 入門
-  Stateflow 入門
-  電気回路シミュレーション入門
-  Power Systems Simulation Onramp

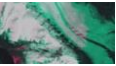

-  パワーエレクトロニクスシミュレーション入門
-  Simulink による制御設計入門

オンライントレーニングスイートに含まれるコース



MATLAB/Simulink

-  MATLAB 基礎
-  Simulink 基礎
-  MATLAB プログラミングアドバンスド
-  MATLAB による複雑なデータの読み込みと前処理および可視化

画像処理および信号処理

-  MATLAB による画像処理
-  Signal Processing with MATLAB

AI、機械学習、ディープラーニング

-  MATLAB によるディープラーニング
-  MATLAB による機械学習

計算数学

-  MATLAB による非線形方程式の解法
-  MATLAB による微分方程式の解法
-  MATLAB によるシンボリック計算
-  MATLAB による線形代数
-  MATLAB による統計解析

※タイトルが英語のものは、英語のみのご用意