

知能機械設計演習

Practicum in Intelligent Machine Design  
MATLAB/Simulinkによる画像処理・機械学習

生命体工学研究科

人間知能システム工学専攻

[s-yasukawa@brain.kyutech.ac.jp](mailto:s-yasukawa@brain.kyutech.ac.jp)

安川真輔

Shinsuke Yasukawa

**1. Matlabを用いた機械学習の初歩（線形回帰）**

2. Matlabの操作方法・コーディング方法（アドバンスド）

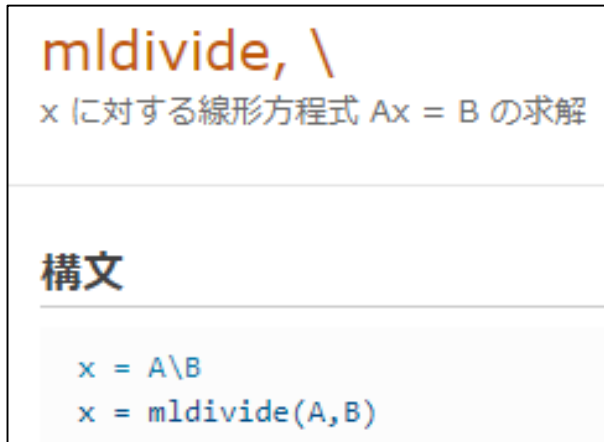
## Matlabを用いた機械学習の初歩（線形回帰を例に） -1/4-

モデル：原因と結果の関係を示すもの

線形回帰：上記の関係が直線的に表せる場合に使う方法

⇒従属(または応答)変数  $y$  と 1 つ以上の独立(または予測子)変数  $x_1, \dots, x_n$  の関係をモデル化する

MATLABプログラムによって、データに基づいてモデルを作り、それをグラフに表示する一連の流れを体験する。



## Matlabを用いた機械学習の初歩（線形回帰を例に） -2/4-

$$y = \beta_0 + \beta_1 x + \epsilon$$

$\beta_0$  : y切片

$\beta_1$  : 勾配（回帰係数）

$\epsilon$  : 誤差項

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}.$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, B = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}.$$

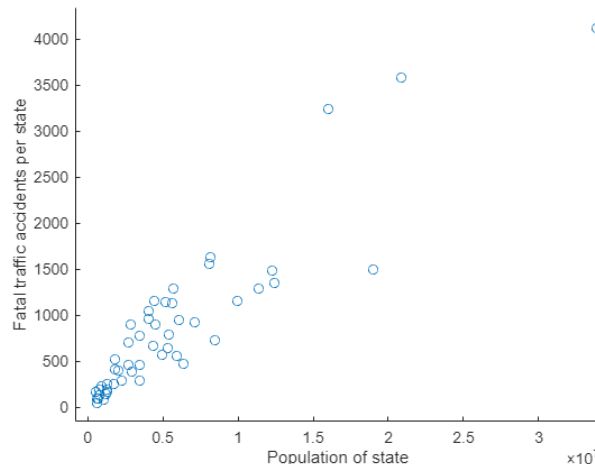
$$Y = XB$$

Matlabではmldivide演算子を  $B = X \backslash Y$  のように使用してBを求めることが出来る

データセット

Accidents

人口と事故の関係



## Matlabを用いた機械学習の初歩（線形回帰を例に） -3/4-

```
load accidents
x = hwydata(:,14); %Population of states
y = hwydata(:,4); %Accidents per state
format long
b1 = x\y
```

b1 = 1.372716735564871e-04

$$y = \beta_1 x = 0.0001372x$$

```
yCalc1 = b1*x;
scatter(x,y)
hold on
plot(x,yCalc1)
xlabel('Population of state')
ylabel('Fatal traffic accidents per state')
title('Linear Regression Relation Between Accidents & Population')
grid on
```

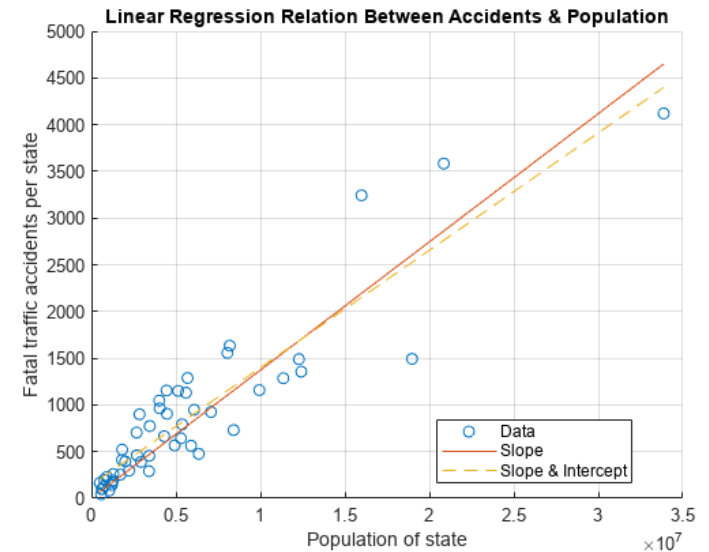


## Matlabを用いた機械学習の初歩（線形回帰を例に） -4/4-

x に 1 の列をパディング

```
X = [ones(length(x),1) x];  
b = X\y  
yCalc2 = X*b;  
plot(x,yCalc2,'--')  
legend('Data','Slope','Slope & Intercept','Location','best');
```

$$y = \beta_0 + \beta_1 x = 142.7120 + 0.0001256x$$



## その他の機械学習チュートリアル

### 分類

バギング分類木を使用した手書き認識  
分類

ウェーブレットベースの機能とサポートベ  
クターマシンを使用した信号分類

ロジスティック回帰モデルのベイズ解析

不均衡なデータによる分類

ベイズ最適化による自動分類器選択

### 回帰

ブースティングされた決定木による回帰:  
ニューヨーク市の住宅価格

重み付けされた非線形回帰

一般化線形モデルを使用したデータの当て  
はめ

### クラスタリング

クラスター評価

クラスター分析

アヤメの花のクラスタリング

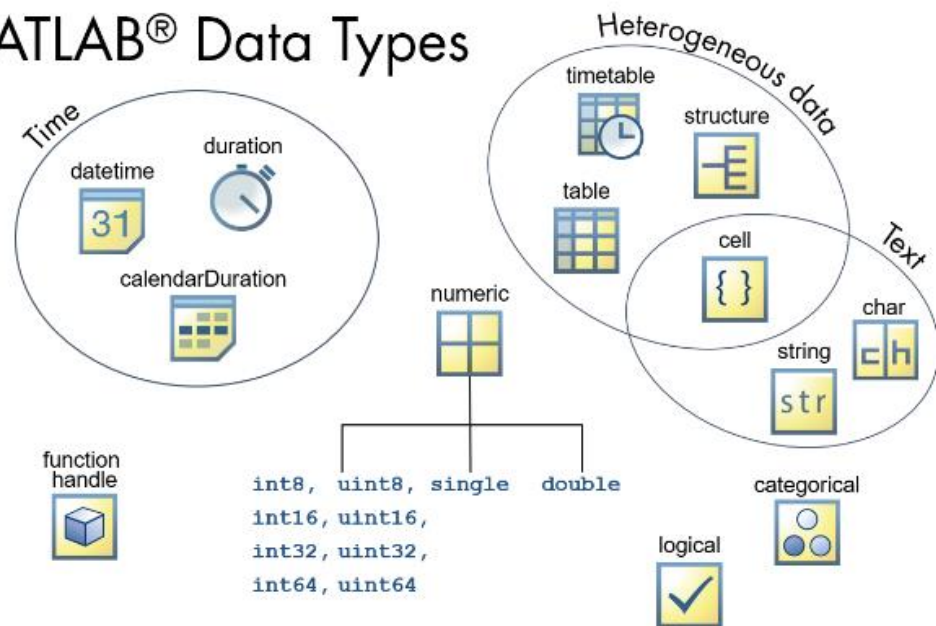
1. Matlabを用いた機械学習の初歩（線形回帰）
- 2. Matlabの操作方法・コーディング方法（アドバンスド）**



- ・データの構造化
- ・異種混合データの操作
- ・コードの最適化
- ・柔軟性の高い関数の作成
- ・ロバストなアプリケーション
- ・アプリケーションの動作と検証
- ・コードのデバッグ
- ・プロジェクトの整理

# データの構造化 1

## MATLAB® Data Types



○データにはさまざまな形式がある。

ex.

数値データ：行列

true と false の値：logical 配列

日付と時刻：datetime 配列

○データの格納方法が複数ある場合もある。

テキスト データ：文字配列 or string, 場合によってはカテゴリカル変数

これらはすべて、各要素が同じクラスである "同種" のデータ型

○異種なデータの組み合わせの場合

→table、timetable、cell 配列、または構造体配列に格納できる。

どれが最適かは、使うデータの種類によって変わる。

(コンテナ変数)

## データの構造化 2

どの異種混合データ型を使用すべきか？

ex. 衝突試験データを解析している場合

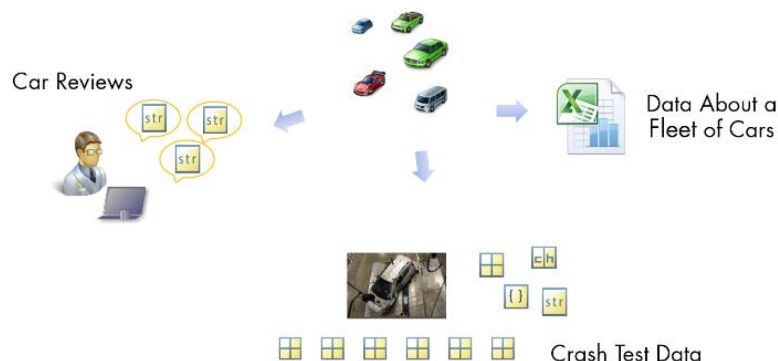
複数のタイプの情報がある。

センサーの数値データ、衝突後の車両のイメージ、  
車両のメーカーとモデルなど

関連する情報をまとめて格納しつつ、

その情報に最も合うデータ型を保持すると便利

(数値には数値データ型、タイムスタンプには **datetime** 配列など)



**table**、**cell** 配列、構造体は、異種混合データを格納する **MATLAB** データ型これらのデータ型のうちどれを選択するかは、個々のデータセットとその使用方法による。

Tables/Timetables

Make	Model	Year	Type	Weight	Length	Width

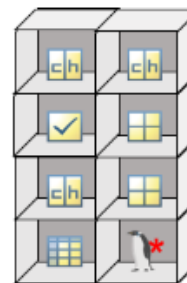
Tabular data with named columns that may have different types

Structures

```
crashData
├── car <1x18>
├── testInfo <2x1>
├── impactSpeed <1x1>
├── data <1435x7>
├── photos <5184x3456x3>
└── ...
```

Heterogeneous data with named fields

Cell Arrays



Heterogeneous data, indexed numerically

## データの構造化 3

配列の型	目的とする内容
single double	小数点以下最大 8 桁 (single) または最大 16 桁 (double) の精度を持つ浮動小数点数 特に指定しない限り、MATLAB ではすべての数値変数を double 型として格納します。
int* uint*	さまざまな範囲の整数値
char string	テキスト
datetime duration calendarDuration	日付、時刻、日付間および時刻間の期間
logical	論理値 (真/偽)
categorical	英数字 (通常はテキスト) のラベルの有限集合に含まれる値
function_handle	関数の定義

## データの構造化 -演習-

```
C = {'one','two','three'; 100,200,rand(3,3)}
```

```
C2 = C(1:2,1:2)    : サブセット
```

```
R = C{2,3}         : 中身へアクセス
```

```
C{1,3} = 'longer text in a third location'
```

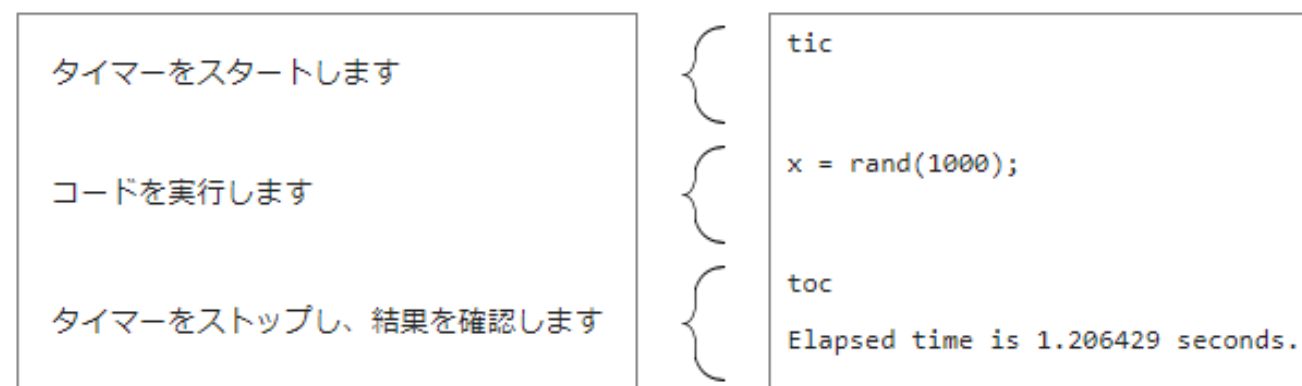
```
C(1,1:2) = {'first','second'}
```

複数の **cell** の内容を同時に置換するには、小かっこを使用して **cell** を参照し、中かっこを使用して同等のサイズの **cell** 配列を定義する。

# コードの最適化

## パフォーマンスの測定とボトルネックの検出

MATLAB コードの実行時間を計測するには、関数 `tic` および `toc` を使用します。

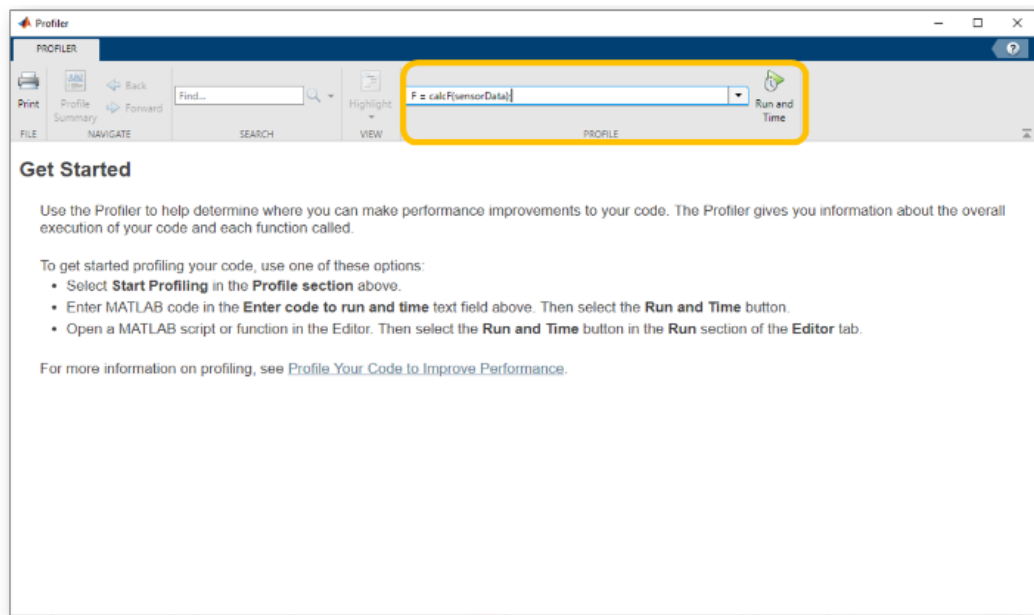


コードのパフォーマンスを改善するには、コードで最も時間がかかっている箇所を把握します。  
MATLAB プロファイラーは、コードのどの箇所で最も時間がかかっているかを分析します。

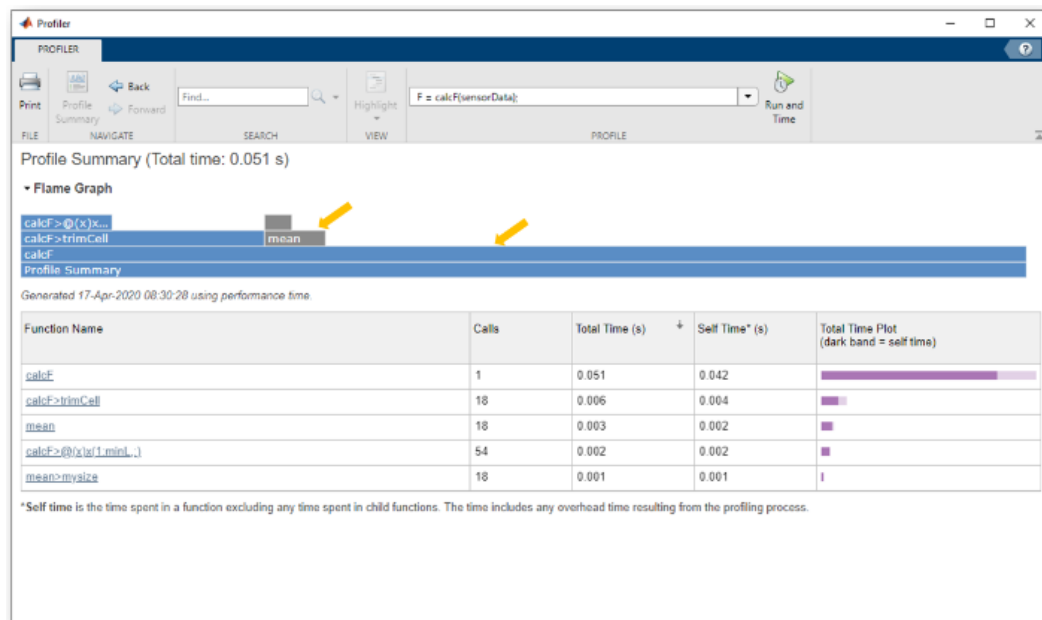
### ▼ Lines that take the most time

Line Number	Code	Calls	Total Time (s)	% Time	Time Plot
13	<code>F(kk) = sqrt(Fx(kk)^2 + Fy(kk)^2 + Fz(kk)^2);</code>	73477	0.009	17.2%	
5	<code>s = trimCell(sensors(k,:));</code>	18	0.007	14.6%	
14	<code>end</code>	73477	0.006	11.2%	
17	<code>Fmean = mean(F);</code>	18	0.004	7.3%	
16	<code>[Fmax(k), maxIdx(k)] = max(F);</code>	18	0.003	5.4%	
All other lines			0.023	44.4%	
Totals			0.051	100%	

# コードの最適化

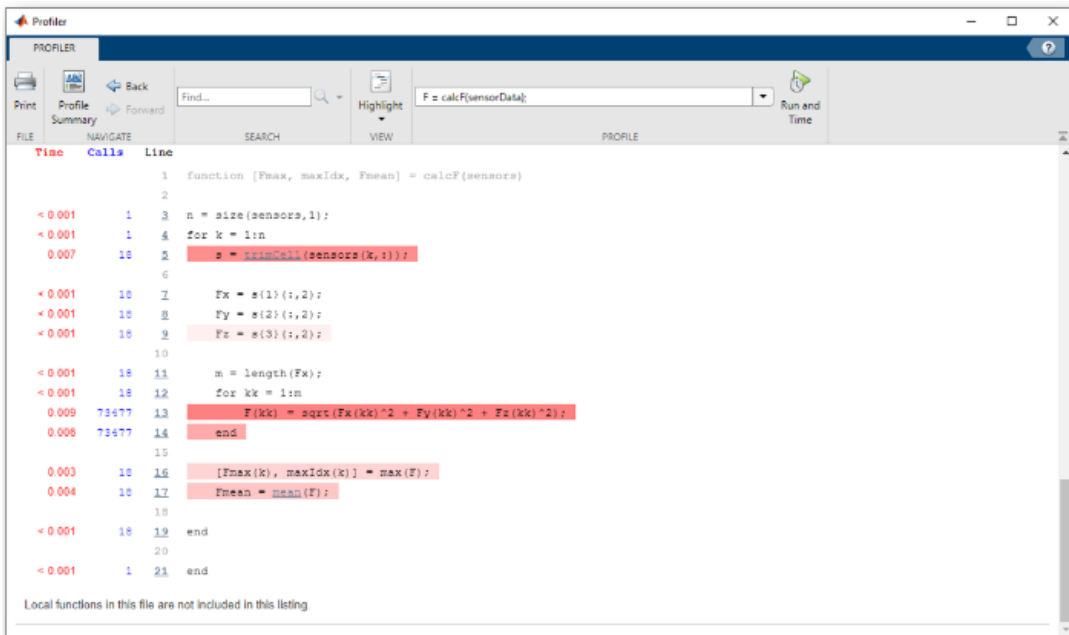
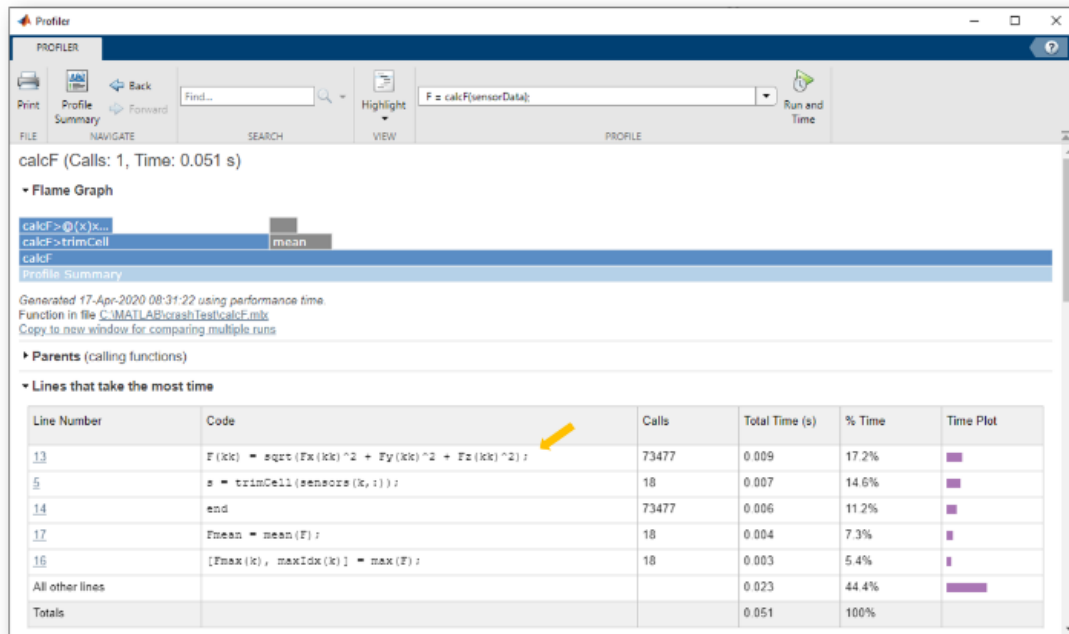


MATLAB プロファイラー  
MATLAB コマンド プロンプト:  
「profile viewer」  
→ローカルのみ



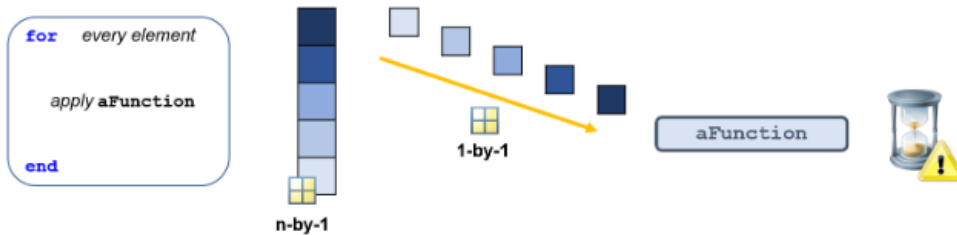
フレーム グラフのグレーの線  
各 MATLAB 関数に対応  
青の線  
プロファイラーおよび  
ユーザー コードでの所要時間

# コードの最適化





# ベクトル化



## for ループを使用する場合

nonVectorized.mlx

```

1. % Generate data
2. r = rand(1,5000);
3.
4. % Compute the difference between
5. % the adjacent elements
6. d = zeros(1,4999);
7. for i=1:4999
8. d(i) = r(i+1)-r(i);
9. end

```

## ベクトル化を使用する場合

vectorized.mlx

```

1. % Generate data
2. r = rand(1,5000);
3.
4. % Compute the difference between
5. % the adjacent elements
6. d = diff(r);

```

## Element-wise Operations

+	-	.
.*	./	.^

## Mathematical Functions

<b>sin</b>	<b>exp</b>	<b>sqrt</b>
<b>abs</b>	<b>round</b>	etc.

## Statistical Functions

<b>sum</b>	<b>mean</b>	<b>max</b>
<b>std</b>	<b>prod</b>	<b>diff</b>
<b>cumsum</b>	<b>cumprod</b>	etc.

## Set Operations

<b>union</b>	<b>intersection</b>	<b>unique</b>
<b>setdiff</b>	<b>setxor</b>	<b>ismember</b>

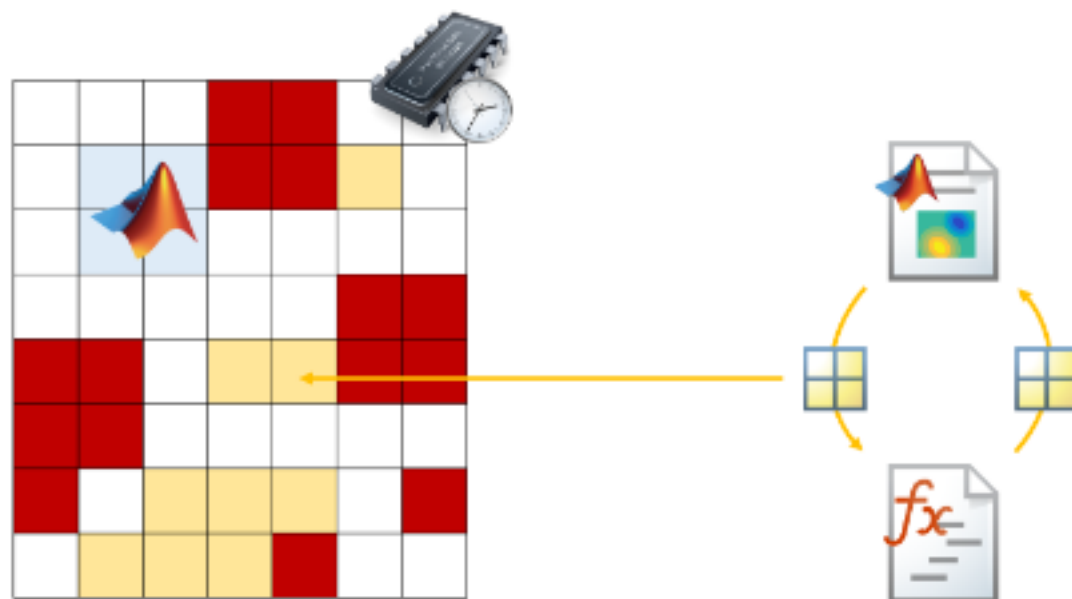
## String Operations

<b>strcmp</b>	<b>strncmp</b>	<b>regexp</b>
<b>strcat</b>	<b>strvcat</b>	<b>cellstr</b>
<b>deblank</b>	<b>lower</b>	etc.

## Logical Functions

<b>is*</b>	<b>any</b>	<b>all</b>
<b>nnz</b>	<b>find</b>	

## メモリについて



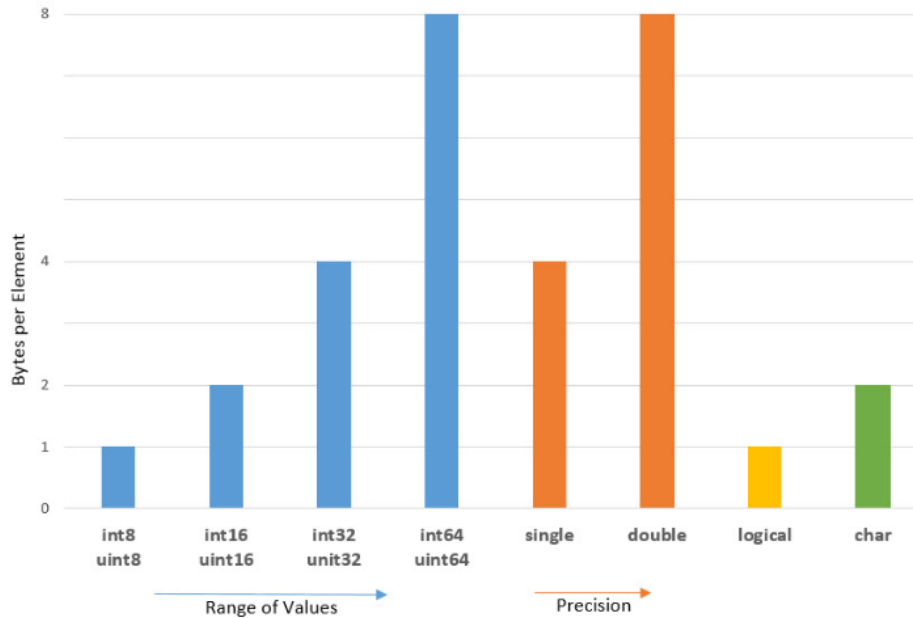
メモリ使用量を改善すると、コードの実行を高速化できるほか、メモリの使用量を抑えることができます。MATLAB データ型に必要なメモリ量を理解すると、コードの効率性を高めるデータ型を選択できます。

# メモリについて

## ○配列より + $\alpha$ の量メモリを消費

### 必要なメモリ量が決まっているデータ型

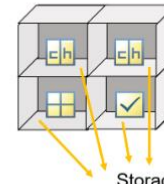
一部の同種のデータ型は、要素ごとに決まったメモリ量を必要とします。



データ型で必要とされるメモリ量は、そのデータ型が表現できる値の範囲または精度に伴って増加します。

```
car = {'Ford', 'Expedition'; 32.7, true}
```

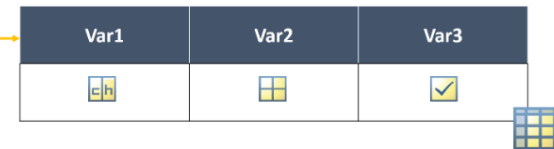
2-by-2



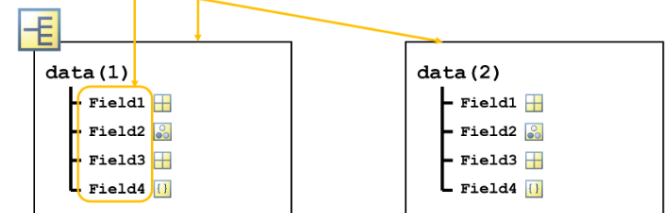
Storage Overhead  
Per cell array element



Storage Overhead



Storage Overhead  
Per field and element

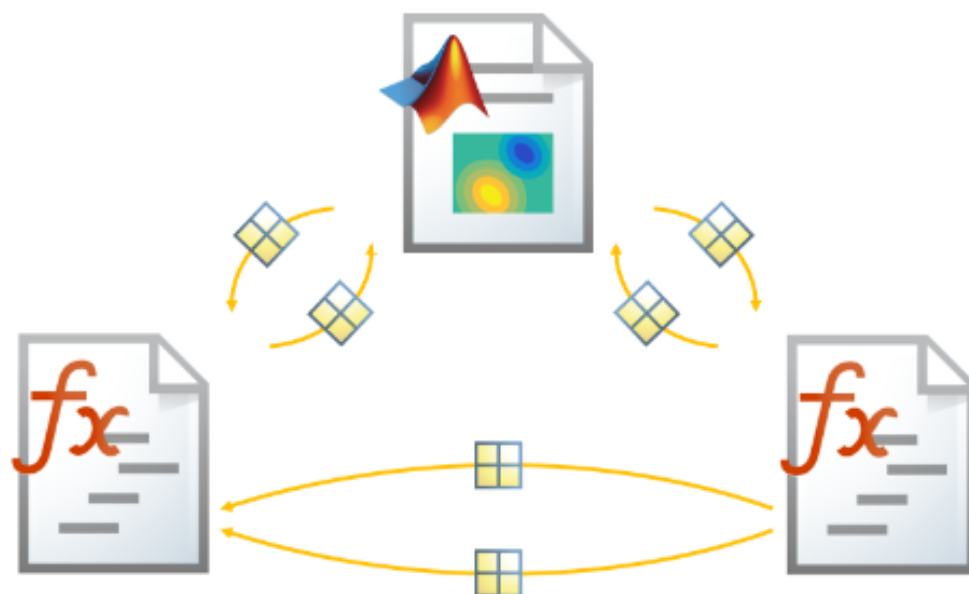


## ○配列の事前割り当て (zeros, onesなど) で高速化

# メモリについて

## 関数でのメモリ使用量の改善

相互に呼び出しを行うスクリプトおよび関数では、多くの場合、データを共有する必要があります。通常、関数には独自のワークスペースがあるため、メモリ管理方法をよく検討して決定すると、効率のよいコードを作成しやすくなります。

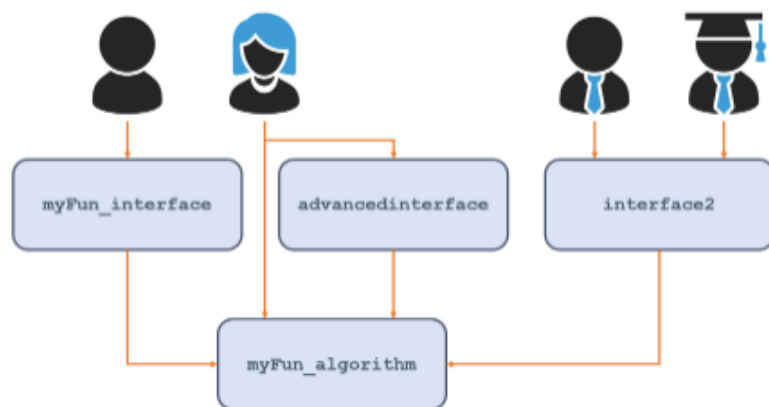


関数に入力を渡すことは、関数間でのデータ共有方法の 1 つです。大きな変数が各ワークスペースに重複して存在する場合は、メモリの問題が発生するおそれがあり、最適化するよい機会となります。

# 柔軟性の高い関数

## 複数のインターフェイスの作成

同じアルゴリズムに基づいた複数のラッパー関数を使用します。それぞれのラッパー関数により、用途ごとに異なるインターフェイスをユーザーに提示できます。



ほとんどのユーザーは `interface2` を使用できます。上級ユーザーは `advancedInterface` を使用する (またはアルゴリズムを直接呼び出す) ことができます。

# 柔軟性の高い関数

## 既定の入力の設定

関数呼び出しですべての入力が指定されたかどうかを確認するには、関数 `nargin` を使用します。関数の本体内で呼び出された場合、`nargin` は関数呼び出し時に渡された入力数を返します。

 `nargin` 入力引数の数

ユーザーに特定の入力だけを指定させるには、空の配列 `[]` を使用します。

- 関数本体内の場合 - 関数 `isempty` を使用して入力为空であるかどうかを確認する。空である場合は、既定値を代入する。
- 関数呼び出し内の場合 - 入力値の代わりに空の配列 `[]` を使用する。

関数呼び出し	エディター
<pre>loanint = interest(500,[],3.1,"compound")</pre>	<pre>function int = interest(p,n,r,type) if isempty(n)     n = 12; end</pre>

# 柔軟性の高い関数

## 可変長の入力引数

関数定義では、複数の入力引数 (名前と値のペアの集合など) を 1 つの入力変数 `varargin` にマッピングできます。

関数呼び出し	エディター
<code>analyzeAndPlot(time,position,"LineWidth",3,"Color","r")</code>	<code>function analyzeAndPlot(x,y,varargin)</code>

# 柔軟性の高い関数

## テキスト入力のマッチング

ユーザーに過剰な制約を課すことなく、ユーザー入力を厳密に期待どおりのもの (型と先頭の大文字/小文字の違いを含む) にするには、`validatestring` を使用します。入力テキストが有効な文字列のいずれとも一致しなかった場合、`validatestring` はエラーを生成します。

```
str = validatestring("distribution",["DistanceMetric","DistributionName","NumSamples"])
str =
    "DistributionName"
str = validatestring("dist",["DistanceMetric","DistributionName","NumSamples"])

Expected input to match one of these values:
"DistanceMetric", "DistributionName", "NumSamples"
The input, dist, matched more than one valid value.

str = validatestring("Algorithm",["DistanceMetric","DistributionName","NumSamples"])

Expected input to match one of these values:
"DistanceMetric", "DistributionName", "NumSamples"
The input, "Algorithm", did not match any of the valid values.
```



# 柔軟性の高い関数

## 可変出力数

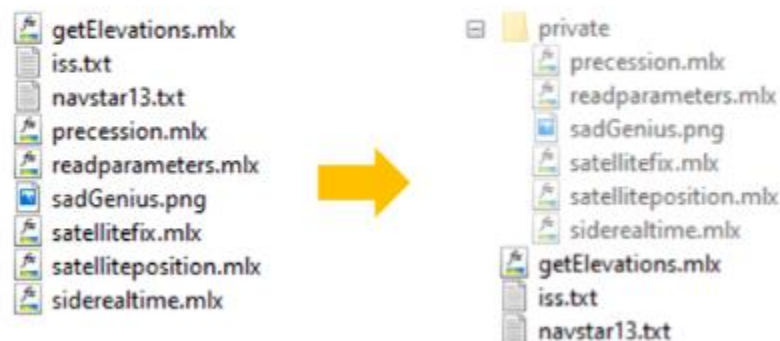
可変の入力数を扱う場合と同じように、`nargout` を使用すると、関数呼び出し時に、要求された出力の数に することができます。また、`varargout` を使用して、任意の数の出力引数を表すこともできます。

関数呼び出し	エディター
<pre>[x,y,z] = myfun(1,2,3)</pre>	<pre>function [a,varargout] = myfun(p,q,r)</pre>

# ロバストなアプリケーションの作成

## プライベート関数

アプリケーションの内部関数をアプリケーションの外部からアクセスできないようにするには、それらの関数を `private` フォルダに配置します。



`private` フォルダは、`private` という名前の通常のフォルダです。

# ロバストなアプリケーションの作成

## ローカル関数

ローカル関数を作成するには、そのコードを同じコード ファイル内の main 関数の下に配置します。

```
findShapes.mlx
1. function findShapes %Primary function
2. ...
3. end
4.
5. function detectCircle %Local function
6. ...
7. end
8.
9. function detectSquare %Local function
10. ...
11. end
```

# ロバストなアプリケーションの作成

## カスタムの警告とエラーの作成

関数 `warning` と関数 `error` を使用すると、カスタムの警告とエラーを生成できます。

**fx** `warning` カスタム警告メッセージを生成します

**fx** `error` カスタム エラー メッセージを生成します

### エラー

```
error("MyProject:invalidValue",...  
    "The value must be numeric.");
```

The value must be numeric.

### 警告


```
warning("MyProject:ValueOutOfRange",...  
    "The value must be between 1 and 10.")
```

The value was expected to be between 1 and 10.

# ロバストなアプリケーションの作成

## 関数入力の検証

与えられた入力の複数の属性をチェックするには、関数 `validateattributes` を使用します。

 `validateattributes` 与えられた変数の複数の属性をチェックします。

### コマンドウィンドウ

```
int = calculateInterest(500,2,-0.03)
```

```
Expected input to be a scalar with  
value >= 0.
```

### エディター

```
function interest = calculateInterest(p,n,r)  
validateattributes("r","numeric",...  
    {'scalar','>='},0}  
...
```

# ロバストなアプリケーションの作成

## エラーのキャッチと処理

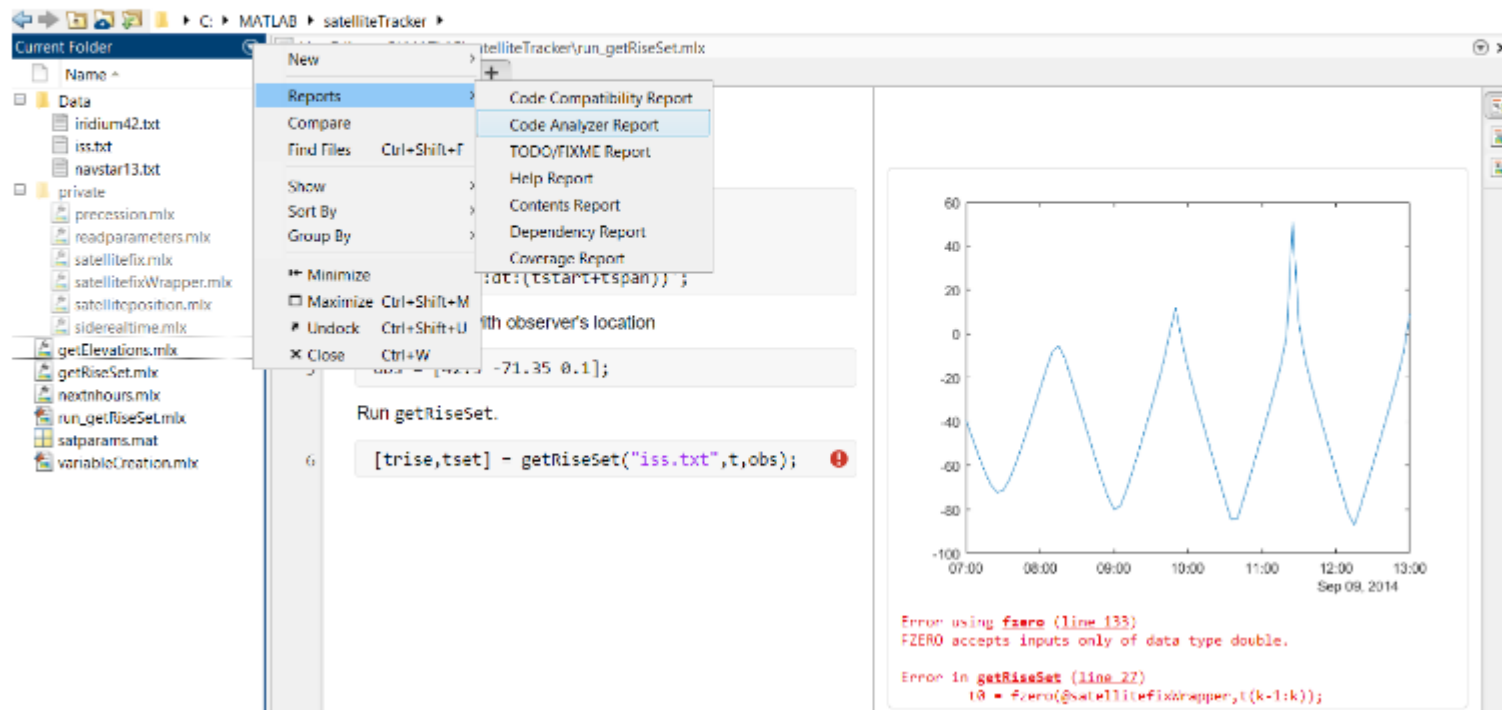
try/catch 構文では、まず try ブロック内のコードが実行されます。エラーが発生すると実行が中止され、直ちに catch ブロックに移ります。

```
try
    % Attempt code
catch mexc
    % Backup case - something went wrong
    % An MException object called mexc now exists
end
```

# プロジェクト管理

## フォルダー レポートの実行

現在のフォルダーに対してフォルダー レポートを実行するには、現在のフォルダーのメニューから [レポート] を選択します。



The screenshot displays the MATLAB IDE interface. On the left, the 'Current folder' pane shows a project structure with files like 'indium42.txt', 'iss.txt', 'navstar13.txt', and various '.mlx' files. The 'Reports' menu is open, showing options such as 'Code Compatibility Report', 'Code Analyzer Report', 'TODO/FIXME Report', 'Help Report', 'Contents Report', 'Dependency Report', and 'Coverage Report'. The 'Code Analyzer Report' is highlighted. The main workspace shows a script 'run\_getRiseSet.mlx' with the following code:

```
Run getRiseSet.  
[trise,tset] = getRiseSet('iss.txt',t,obs);
```

Below the code, a plot shows satellite elevation data. The x-axis is labeled 'Sep 08, 2014' and ranges from 07:00 to 13:00. The y-axis ranges from -100 to 60. The plot shows a blue line representing elevation over time, with a significant peak around 11:00.

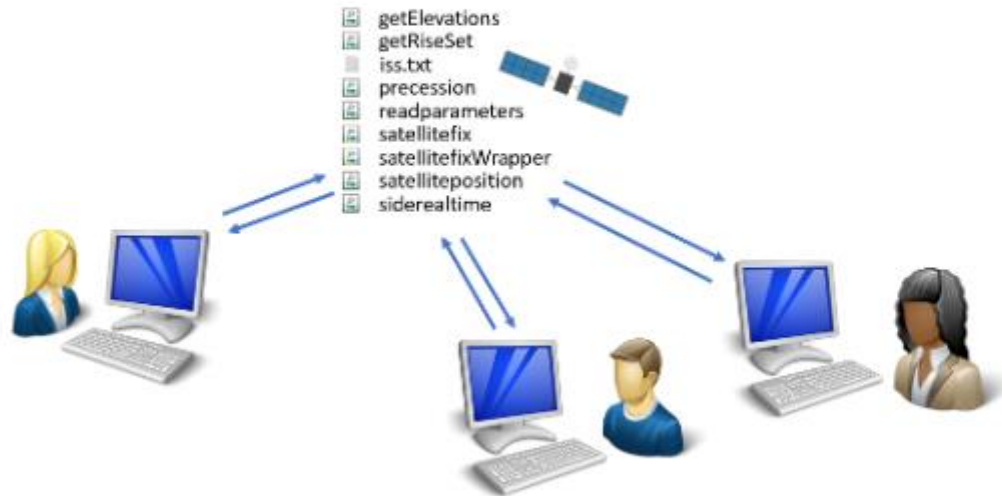
Error messages are displayed at the bottom:

```
Error using fzero (line 133)  
FZERO accepts inputs only of data type double.  
  
Error in getRiseSet (line 22)  
t0 = fzero(@satellitefixWrapper,t(k-1:k));
```

# プロジェクト管理

## MATLAB プロジェクト

MATLAB プロジェクトは、複数のファイルにまたがるコードを管理するのに役立ちます。また、他の人とコードを共有する場合にも使用します。





# アプリケーションの動作と検証

## ソース管理

ソース管理を使用すると、プロジェクトの以前のバージョンを追跡したり管理したりできます。

