MATLAB/Simulink 最新情報

山本 順久
Focus  Collaboration  Production
Explore data
Plot some of the events on a map and use the size of the bubble as the damage cost.

```r
threshold = 7648 ;
plotEventCostsMap(data, threshold);
```

Event frequency by location
Calculate damage costs by group.

```r
damageByEvent = groupsummary(data,"weathercats",["mean","std"],"damage_total")
```

<table>
<thead>
<tr>
<th>weathercats</th>
<th>GroupCount</th>
<th>mean_damage_total</th>
<th>std_damage_total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avalanche</td>
<td>39</td>
<td>1.2500</td>
<td>5.5160</td>
</tr>
<tr>
<td>Blizzard</td>
<td>961</td>
<td>4.8792</td>
<td>38.3841</td>
</tr>
<tr>
<td>Coastal Weather</td>
<td>6053</td>
<td>356.5849</td>
<td>1.1945e+04</td>
</tr>
<tr>
<td>Debris Flow</td>
<td>318</td>
<td>111.1654</td>
<td>736.8761</td>
</tr>
<tr>
<td>Dense Fog</td>
<td>1115</td>
<td>0.8292</td>
<td>9.3457</td>
</tr>
<tr>
<td>Dense Smoke</td>
<td>61</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Drought</td>
<td>3584</td>
<td>0.1853</td>
<td>2.4966</td>
</tr>
<tr>
<td>Dust Devil</td>
<td>14</td>
<td>41.5714</td>
<td>132.2025</td>
</tr>
<tr>
<td>Dust Storm</td>
<td>151</td>
<td>8.8298</td>
<td>48.1969</td>
</tr>
</tbody>
</table>
Fill Missing Data

To replace NaN values in the data and visualize the results, open the Clean Missing Data task. Start by typing the keyword missing in a code block, and then click Clean Missing Data when it appears in the menu.

Select the input data and the cleaning method:

**Clean Missing Data**

- **cleanedData** = Filled missing data in

**Select data**
- **Input data**: cleanedData2
- **X axis**: default

**Specify method**
- **Cleaning method**: Fill missing
- **Moving window**: Centered
- **Max gap to fill**: 0.06

**Display results**
- **Cleaned data**
- **Filled missing entries**
Create Plot
Create a plot interactively

Select visualization
Search for a visualization Filter by Category All

Select data
X default
Y select

Select optional visualization parameters
Create Plot
Clean Missing Data
Clean Outlier Data
Find Change Points
Find Local Extrema
Remove Trends
Smooth Data

Live Editor Tasks make coding optional

DATA AND VISUALIZATION
- Create Plot

DATA PREPROCESSING
- Clean Missing Data
- Clean Outlier Data
- Find Change Points
- Find Local Extrema
- Remove Trends
- Smooth Data

TABLES AND TIMETABLES
- Join Tables
- Retime Timetable
- Stack Table Variables
- Synchronize Timetables
- Unstack Table Variables

OPTIMIZATION
- Optimize

CONTROL SYSTEM DESIGN AND ANALYSIS
- Convert Model Rate
- Reduce Model Order
- Tune PID Controller

PREDICTIVE MAINTENANCE
- Estimate Approximate Entropy
- Estimate Correlation Dimension
- Estimate Lyapunov Exponent
- Extract Spectral Features
- Reconstruct Phase Space

SYSTEM IDENTIFICATION
- Estimate Process Model
- Estimate State-Space Model

SIGNAL PROCESSING AND COMMUNICATIONS
- Extract Audio Features

SYMBOLIC MATH
- Simplify Symbolic Expression
- Solve Symbolic Equation

IMAGE ACQUISITION
- Acquire Webcam Image
Edit at the Speed of Thought
This implements the electromechanical components of a Faulhaber Series 0615 DC-Micromotor permanent magnet electric motor.

The containing system needs to provide voltage and mechanical load.
Simulink runs faster
out-of-the box
Parallel Execution
10 seconds

Thread 1
Block 1  Block 2  Block 3

Thread 2
Block 4  Block 5

Thread 3
Block 6  Block 7  Block 8
Parallel Execution
≈ 3x Speedup

Model Reference

S-Function

Co-Simulation FMU
Simulink Cache
You

Team Members

Source Control System

Design changes

CI System & Build Archive

Simulink cache files
Upper Bounds

For Each Subsystem
SIMD: Single Instruction Multiple Data

\[
\begin{align*}
A_1 & + B_1 = C_1 \\
A_2 & + B_2 = C_2 \\
A_3 & + B_3 = C_3 \\
A_4 & + B_4 = C_4
\end{align*}
\]

Scalar operation
SIMD operation
Average Speedup in Customer Workflows

- R2015a: 1x
- R2016a: 1.49x
- R2017a: 1.72x
- R2018a: 2.06x
- R2019a: 2.12x
- R2020a: 2.18x
- R2021a: 2.23x
<table>
<thead>
<tr>
<th>Area</th>
<th>Description</th>
<th>Performance Improvement</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphics</td>
<td>rendering performance, large data sets in UI figures</td>
<td>6x</td>
<td>R2020a</td>
</tr>
<tr>
<td>Area</td>
<td>Description</td>
<td>Performance Improvement</td>
<td>Release</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Graphics</td>
<td>rendering performance, large data sets in UI figures</td>
<td>6x</td>
<td>R2020a</td>
</tr>
<tr>
<td>Indexing</td>
<td><code>datetime</code>, <code>duration</code>, and <code>calendarDuration</code> arrays</td>
<td>25x</td>
<td>R2020a</td>
</tr>
<tr>
<td></td>
<td><code>table</code> arrays</td>
<td>2x</td>
<td>R2020a</td>
</tr>
<tr>
<td>Area</td>
<td>Description</td>
<td>Performance Improvement</td>
<td>Release</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Graphics</td>
<td>Rendering performance, large data sets in UI figures</td>
<td>6x</td>
<td>R2020a</td>
</tr>
<tr>
<td>Indexing</td>
<td><code>datetime</code>, <code>duration</code>, and <code>calendarDuration</code> arrays</td>
<td>25x</td>
<td>R2020a</td>
</tr>
<tr>
<td></td>
<td><code>table</code> arrays</td>
<td>2x</td>
<td>R2020a</td>
</tr>
<tr>
<td>Sparse</td>
<td>Matrix multiplication linear systems</td>
<td>4-5x</td>
<td>R2021a</td>
</tr>
</tbody>
</table>
Performance

- Sparse Matrix Multiplication: Improved performance multiplying large sparse matrices
- Sparse Linear Systems: Improved performance solving sparse linear systems $A^tX = B$ with multicolm B
MATLAB Release Notes

R2021a

Performance

- **Sparse Matrix Multiplication:** Improved performance multiplying large sparse matrices
- **Sparse Linear Systems:** Improved performance solving sparse linear systems $A\times X = B$ with multicolomn $B$
**Performance**

- **Sparse Matrix Multiplication**
- **Sparse Linear Systems**

Improved performance multiplying large sparse matrices

Improved performance solving sparse linear systems \(A^TX = B\) with multicolumn \(B\)

Solving a linear system of the form \(A^TX = B\) by executing \(X = A\backslash B\) shows improved performance when \(A\) is a sparse square matrix and \(B\) is a matrix with two or more columns. The speedup applies to the solving step of the calculation but not the factorization step. The performance improvement arises from added support for multithreading, and therefore the speedup gets better as the number of columns in \(B\) increases.

For example, if you solve \(A^TX = B\) using a 1e4-by-1e4 sparse coefficient matrix with approximately 40,000 nonzeros and a \(B\) matrix with 100 columns, performance in R2021a is about \(5x\) faster than in R2020b on a machine with 6 physical cores. This code uses decomposition to factor the coefficient matrix, so only the solving process is timed. If you use \(X = A\backslash B\) instead, you still see a speedup, but the time required to factor the matrix is included and has not changed.

```matlab
function timingSparseBackslashMultRHS
    rng default
    A = sprand(1e4,1e4,0.0003) + speye(1e4);
    B = sprand(1e4,100,0.002);
    dA = decomposition(A);
    tic
    x = dA\B;
    toc
end
```

The approximate execution times are:

- R2020b: 1.5 s
- R2021a: 0.3 s

The code was timed on a 6-core test system by calling the function `timingSparseBackslashMultRHS`.
Deep Learning

<table>
<thead>
<tr>
<th>Area</th>
<th>Description</th>
<th>Performance Improvement</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>Multi-GPU</td>
<td>1.6x</td>
<td>R2020cR2020b</td>
</tr>
</tbody>
</table>

![Graph showing performance improvement across different GPU configurations]
# Deep Learning

<table>
<thead>
<tr>
<th>Area</th>
<th>Description</th>
<th>Performance Improvement</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>Multi-GPU</td>
<td>1.6x</td>
<td>R2020a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>R2020b</td>
</tr>
<tr>
<td>Inference</td>
<td>GPU</td>
<td>2.8x</td>
<td>R2018b</td>
</tr>
<tr>
<td></td>
<td>CPU</td>
<td>2.5x</td>
<td>R2021a</td>
</tr>
</tbody>
</table>

![Graph showing performance improvement across multi-GPU setups]
readtable("myfile.xlsx", "TextType", "string", "Encoding", "UTF-8")
str = ["String was introduced in R2016b."
    "Pattern was added in R2020b."];
str = ["String was introduced in R2016b.
  " Pattern was added in R2020b."];

Create a pattern to match releases

pat = "R" + digitsPattern(4) + ("a"|"b");
```matlab
str = ['String was introduced in R2016b."
    " Pattern was added in R2020b.'];

Create a pattern to match releases

pat = 'R' + digitsPattern(4) + ('a'|'b');

Extract the releases that were mentioned

extract(str,pat)

ans = 2x1 string
    "R2016b"
    "R2020b"
```
mSend

Send Component

mReceive

Receive Component

In
How do you...
Select signals to display
Get insights from simulations quickly without writing code
Tune and monitor simulations without going into the details
\begin{verbatim}
P = 1:40; 
T = 350; 
gas = "carbon dioxide"; 
"ammonia" 
Tcrit : "argon"; 
Pcrit : "butane" 
R = 8; 
"carbon dioxide" 
a = 27; 
b = R*; 
Z = zel; 
for i : "ethylene" 
Val = "fluorine" 
\end{verbatim}
Turn a script into a simple app in seconds
Estimating Sunrise and Sunset

We can estimate sunrise and sunset times if we know the latitude, longitude, and UTC offset. We need to calculate two values:

- Solar time correction
- Solar declination

The solar time correction is the difference (in minutes) between solar time and local time.

The solar declination ($\delta$) is the angle of the sun relative to the earth's equatorial plane. On any given day of the year ($d$), solar declination ($\delta$) can be calculated from the following formula:

$$\delta = \sin^{-1} \left( \sin(23.45) \sin \left( \frac{360}{365} (d - 81) \right) \right)$$

Using the latitude ($\phi$), the sun's declination ($\delta$) and the solar time correction (SC) we can calculate sunrise and sunset times.

$$\text{sunrise} = 12 + \cos^{-1} \left( -\tan(\phi) \tan(\delta) \right) \cdot \text{SC}$$

$$\text{set} = 12 + \cos^{-1} \left( -\tan(\phi) \tan(\delta) \right) \cdot \frac{\text{SC}}{60}$$

Estimating the Sunrise and Sunset Times

Set the latitude, longitude, and UT offset. Notice what happens to the sunrise and sunset times when the latitude is more than 86 degrees N or S (within the polar circles):

\[
\begin{align*}
\text{lat} &= 42 \\
\text{lon} &= -71 \\
\text{UTCoff} &= 0.5
\end{align*}
\]

Estimate the sunrise and sunset times. We use the custom \texttt{equationOfTime} function to calculate the solar time correction (SC).

\[
\begin{align*}
day &= 1.365 \\
\text{timeCorr} &= \text{equationOfTime}(day) \\
\text{solarCorr} &= 4 \times (\text{lon} + 15 \times \text{UTCoff}) + \text{timeCorr} \\
\delta &= \sin \left( \sin(23.45) \sin \left( \frac{360}{365} (\text{day} - 81) \right) \right) \\
\text{sunrise} &= 12 - \cos^{-1} \left( -\tan(\text{lat}) \tan(\delta) \right) \times 365 \times \text{solarCorr} / 60
\end{align*}
\]
Estimating Sunrise and Sunset

We can estimate sunrise and sunset times if we know the latitude, longitude, and UTC offset. We need to calculate two values:

- Solar time correction
- Solar declination

The solar time correction is the difference (in minutes) between solar time and local time.

The solar declination ($\delta$) is the angle of the sun relative to the earth's equatorial plane. On any given day of the year ($d$), solar declination ($\delta$) can be calculated from the following formula:

$$\delta = \sin^{-1} \left( \sin(23.45) \sin \left( \frac{360}{365} (d - 81) \right) \right)$$

Using the latitude ($\phi$), the sun's declination ($\delta$) and the solar time correction (SC) we can calculate sunrise and sunset times.

$$\text{sunrise} = 12 - \frac{\cos^{-1} \left( -\tan \phi \tan \delta \right)}{15} \cdot \frac{SC}{60}$$

$$\text{sunset} = 12 + \frac{\cos^{-1} \left( -\tan \phi \tan \delta \right)}{15} \cdot \frac{SC}{60}$$

Estimating the Sunrise and Sunset Times

Set the latitude, longitude, and UT offset. Notice what happens to the sunrise and sunset times when the latitude is more than 66 degrees N or S (within the polar circles).

1. $\text{lat} = 42$
2. $\text{lng} = -71$
3. $\text{UTCoFF} = 5$

Estimate the sunrise and sunset times. We use the custom `equationOfTime` function to calculate the solar time correction (SC).
t = -pi:pi/20:pi;
comet(t,tan(sin(t))-sin(tan(t)))
Languages

C/C++
Python
Java

Development

Operations
**Languages**
- C/C++
- Python
- Java

**Source Control & CI**
- GitHub
- Git
- Jenkins
- Travis CI
- CircleCI

**Development**

**Operations**
Source Control & CI
- GitHub
- Git
- Jenkins
- Travis CI
- CircleCi

Languages
- C/C++
- Python
- Java

Data Platforms & Technology
- Domino Data Lab
- Databricks
- Jupyter
- Tableau
- Kafka
- Hadoop
- MQTT
- RabbitMQ

Cloud Platforms & Technology
- AWS
- Azure
- Docker
- Docker Hub
Integrate with Python-Based Frameworks
Deep Learning in Simulink
Simulink is the Simulation Integration Platform
Integrating your C code into Simulink is simple.
Code Importer

- R2021a
- Code Import Successful
  - A Simulink library has been generated in the output folder.
  - Next steps:
    - Open generated Simulink library
    - Change the MATLAB current folder to the output folder.
    - Add generated Simulink library files to project: Add

- Instrument Control Toolbox
  - HDL Coder
  - HDL Verifier
  - Image Acquisition Toolbox
  - MBD Tools
  - Mixed-Signal Blockset
  - Model Predictive Control
  - Motor Control Blockset
  - Navigation Toolbox
  - OPC Toolbox

- <FunctionName>
- C Caller
- C Function
Flexibility to simulate and co-simulate

FMU Import and Export
Branch 1

Branch 2
Simulink.exportToVersion(a, b, c)

R2021a

Up to 7 years

R2014a

Project to export
ZIP file name
Desired release
NonLinearActuator_harness

1. Cmd.
2. Signal spec. and routing
3. NonLinearActuator
5. ACT_BUS
6. Signal spec. and routing
7. ACT_BUS
Use Object-Oriented Programming to Model Real-World Objects

Object-oriented programming is a design approach that enables you to programmatically define structures called objects that combine data (properties) together with functions that operate on that data (methods). In MATLAB®, you can create objects that model the behavior of devices and systems in the real world. Those objects can then be used as building blocks in applications used to simulate and analyze complex systems.

Learn more
- Developing and Deploying Sonar and Echosounder Data Analysis Software
- Building and Extending Portfolio Optimization Models with MATLAB
- Control System Modeling with Model Objects
Country Average Temperatures (1901–2012)
Data courtesy of World Bank Climate Data

About This App

This app was built in MATLAB® using features from R2019a through R2021a. The app’s layout is managed using `UIGRIDLAYOUT`. The side panel is a combination of HTML, JavaScript, and CSS integrated via the `UIMHTML` function, plus `UIIMAGE` for the MathWorks logo. The layout for the legends and the `BUBBLECHART` and `SWARMCHART` visualizations are managed using a flow `TILEDLAYOUT`. Interactivity makes use of default interactions and custom `DataTipTemplate` properties. Lastly, this info panel uses an HTML interpreter for the label’s text, and a `UIHYPERLINK` for the link.
/* Function for Chart: '<Root>/State_Machine' */
static void State_Machine_MainStateMachine(void)
{
    SRE tmp_0;
    boolean_T tmp;
    switch (State_Machine_DW.is_MainStateMachine) {
    case State_Machine_IN_Charging:
        if (State_Machine_DW.FaultPresent) {
            /* Outport: '<Root>/ChargeCurrentReq' */
            State_Machine_Y.ChargeCurrentReq = 0.0F;
        }
        State_Machine_Y.is_Charging = State_Machine_IN_NO_ACTION;
        break;
    case State_Machine_MT_Charging:
        /* Outport: '<Root>/BMS_State' */
        State_Machine_Y.BMS_State = BMS_Fault;
        break;
    }
}
Fill Missing Data

To replace Null values in the data and visualize the results, open the Clean Missing Data task. Start by typing the keyword missing in a code block, and then click Clean Missing Data when it appears in the menu.

Select the input data and the cleaning method:
- Select data
  - Input data: cleanedData
  - X-axis: default

Specify method:
- Cleaning method: Fill missing
- Moving window: Centered
- Max gap to fill: 0.01

Display results:
- Cleaned data
- Filled missing entries

This second-order differential equation characterizes the system:

\[
\ddot{r} = \frac{F - \dot{r} - kr}{m}
\]

Solving for acceleration provides a form of this equation that maps more closely to a Simulink model.
Simulink is the Simulation Integration Platform
MATLAB Online
Use MATLAB and Simulink through your web browser
Start using MATLAB Online

Simulink Online
Use Simulink through your web browser
Start using Simulink Online

Use MATLAB and Simulink with no downloads or installations.

Collaborate with others through online sharing and publishing

Store, manage, and access your files anywhere.

All named academic, commercial and home user accounts are eligible
MATLAB EXPO
2021

Thank you